Scalable Multicast Routing: Protocol Design and Performance Evaluation

by

Filli Y.Y. Cheng

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF PHILOSOPHY
in
Department of Computing
The Hong Kong Polytechnic University

Abstract

Existing multicast routing protocols create either source-specific delivery trees, e.g., Distance Vector Multicast Routing Protocol (DVMRP) and Protocol Independent Multicast-Dense Mode (PIM-DM), or a shared delivery tree for all sources, e.g., Protocol Independent Multicast-Sparse Mode (PIM-SM) and Core Based Tree (CBT). The source-specific routing protocols were designed to achieve efficient end-to-end delay by using a broadcast-and-prune approach. However, this approach does not scale to both the number of sources and the number of multicast groups. The shared-tree protocols, on the other hand, scale to the number of sources in a multicast group by building a delivery tree rooted at a core router that is shared by any sources. Since multicast packets no longer travel along the shortest path to each member in a multicast group, shared trees incur longer end-to-end delay in the delivery of the packets. The inappropriate location of core router can worsen the delay performance. Furthermore, the shared tree is not scalable to the number of multicast group. Although some protocols, such as PIM-SM, allow a co-existence of source-specific trees for some sources and a shared tree for other sources, the problems mentioned above remain for the respective delivery trees.

This thesis addresses two problems to improve the delay and scalability performance of the current multicast routing protocols:

1. How can we obtain a good delay performance as well as a good scalability performance for multicast routing in an effective manner?
2. Given a set of multicast delivery trees, created by some routing protocols, how can we increase their scalability performance?

Adaptive Source and Core Based Multicast

Because all existing multicast routing protocols cannot address the first problem, we propose a novel multicast routing protocol, coined as Adaptive Source and CORe Based MulticasT (ASCORT). ASCORT is neither a pure source-specific approach nor a pure shared tree approach. In fact, an ASCORT delivery tree can be considered a two-level structure. In the lower level, each member host is connected to only one source-specific tree, called a reduced tree; therefore, a reduced tree connects only a subnet of member hosts. In the upper level, the reduced trees are connected in the manner of a shared tree, and one of the source's first hop router is selected as a "core router" for this level. While the delivery tree on the lower level is uni-directional, the one on the upper level is bi-directional.

The way that ASCORT establishes reduced trees ensures that the resultant reduced trees are disjoint. One simple way to achieving that is to use hop counts from the sources as a criterion for establishing the reduced trees. As a result, each member host connects a nearest source's first-hop router, thus minimizing the delay between the root of the reduced tree and the member host. Moreover, the routers on the reduced tree are required to store state information for that particular tree, independent of other reduced trees for the multicast group, thus minimizing the state requirement for each reduced tree. For the bi-directional tree, the root is selected from the sources' first hop routers; experimental results have showed that this selection method achieves better average end-to-end delay as compared with a random method. Extensive simulation studies indeed show that ASCORT achieves good delay as well as good scalability. Besides delay and scalability, ASCORT possesses many other desirable properties for multicast routing:

- ◊ Unlike the shared tree approach, ASCORT does not require an additional protocol for electing a core router.
- ◊ ASCORT is a distributed protocol and it incurs a very low overhead for operating the protocol.
- ◊ ASCORT delivery trees are loop-free given that the underlying unicast routing does not contain loops.
- ◊ ASCORT is independent of the choice of the underlying unicast routing protocol.
- ◊ Unlike some shared-tree routing protocols, ASCORT does not use tunneling techniques, thus facilitating multicast routing with QoS requirement.

Multicast Tree Switching

The current multicast routing protocols fail to scale to the number of multicast groups. There are two possible approaches to this problem. The first one is for a multicast router to perform state aggregation for a number of groups. This approach assumes that there are significant "overlaps" among multicast trees, so that a router may keep states for different multicast groups at the same time. Even when this assumption is valid, multicast state aggregation is not trivial, because, unlike unicast addresses, multicast addressing space is flat. A second approach is to modify the paths of a multicast tree, such that the new path requires fewer states as compared with the original path.

In this thesis, we have proposed a new Tree Switching Protocol (TSP) to implement the second approach, and at the same time to increase the tree overlapping, so that the first approach can be applied to further reduce the multicast state requirement. Specifically, given a forest of multicast trees created by the same multicast routing protocol, say PIM-SM, TSP selects a base multicast tree, which is based on some static information, such as the IP address, or others and it

allows other trees to switch to the base tree. If a candidate tree cross-overs with the base tree on some router, TSP will establish a new tree branch to the base tree, and the old branch will be pruned away. TSP is a distributed protocol and is initiated by a leaf router independently from others. Extensive simulation results have shown that TSP can reduce the amount of multicast state by as much as 30%; with additional state aggregation using leaky and nonleaky methods, the amount of state reduction can be further increased by a significant amount. In addition, the TSP possesses many attractive properties:

- ◊ The process of creating new branches and pruning the old ones does not create routing loops.
- ◊ TSP is independent of the underlying multicast routing protocols.
- ◊ TSP is independent of state aggregation methods.
- ◊ TSP is adaptive to the degree of overlapping existing in a forest of multicast trees.

Acknowledgments

It is my pleasure to express my sincere gratitude to my chief supervisor, Dr. Rocky Chang, for providing me with useful guidance, constructive suggestions, and continuous support in my graduate study at The Hong Kong Polytechnic University.

I would also like to thank Prof. Jennifer Hou and Dr. Xiaohua Jia for being my external examiners. Their insightful comments have significantly improved the readability of this thesis.

Furthermore, I present my heartfelt gratitude to my colleagues for providing me a comfortable and warm study environment.

Last but certainly not least, I thank my parents for their encouragement and support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Multicasting is concerned with the distribution of the same information from a sender to multiple receivers concurrently. The rapid advances in computer technology allow many Internet applications such as distance learning, data replication, and teleconferencing to emerge. These applications require the ability to disseminate the same information from a sender to many receivers simultaneously. In other words, multicast capability is required for them to be realized in the Internet. Among the technology emerged in computer networks, multicast routing is an advanced technology to provide multicast capability in internetworks. It is for this reason that multicast routing is a critical technology for realizing those applications which are of multicast nature.

In order to facilitate the interoperability of different protocols and the scalability of the network, the Internet is organized in the form of a three-level hierarchy. This hierarchy is composed of (i) *LAN level*, (ii) *intra-domain level*, and (iii) *inter-domain level*. The LAN level is composed of a set of local area networks (LANs) each of which has both hosts of users and a *designated router*. A host in a LAN can be a source or a member in a multicast group. Using the Internet Group Membership Protocol (IGMP) [Deering 1989], a designated router is able to communicate with each host in the same LAN. The designated router, which has one or more members in its LAN, is referred as the *member router* whereas the designated router, which has one or more sources in its LAN, is referred as the *first-hop router*. By connecting the designated routers of different LANs together, different domains are formed. These domains, in turn, form the intra-domain level. The domains are then interconnected with routers called *border routers* and the network of domains forms the inter-domain level.

Different multicast routing protocols can be employed at the intra-domain and the inter-domain levels. Such kind of interoperability is provided by using an interface similar to that described in [Thaler, Estrin, and Meyer 1998]. Many of existing multicast routing protocols such as [Aharoni and Cohen 1997; Ballardie 1997; Ballardie, Francis, and Crowcroft 1993; Banerjea, Faloutsos, and Pankaj 1998; Deering 1991; Deering *et al.* 1996; Estrin *et al.* 1995, 1997; Faloutsos, Banerjea, and Pankaj 1998; Floyd *et al.* 1996; Garcia-Luna-Aceves and Madruga 1999; Kompella, Pasquale, and Polyzos 1993; Moy 1994; Shields 1996; Shields and Garcia-Luna-Aceves 1997; Parsa and Garcia-Luna-Aceves 1995, 1997; Rouskas and Baldine 1997; Salama 1996; Tode *et al.* 1992; Waitzman, Partridge, and Deering 1988] are developed for the intra-

domain level. A number of these protocols can be extended to the inter-domain level using hierarchies as shown in [Handley, Crowcroft, and Wakeman 1997; Shields and Garcia-Luna-Aceves 1997; Thyagarajan and Deering 1995].

To distribute packets from sources to members in a multicast group, multicast distribution trees are built and additional forwarding states are maintained at the multicast routers on the trees for multicasting. We call the multicast routers, which are on the multicast distribution tree, as *on-tree routers*. A multicast router is either a *leaf router* or an *interior router* on a multicast distribution tree. The leaf routers terminate the forwarding of multicast packets and the interior routers forward multicast packets to downstream routers.

In general, multicast distribution trees can be classified into two categories. They are *source-specific trees* or *shared trees*. A source-specific tree is rooted at a source and each source has its own source-specific tree to disseminate multicast packets. In a source-specific tree, the shortest path is available between each pair of sources and members. On the other hand, a shared tree rooted at a common router called *core router* or *rendezvous point* and all the sources use this same shared tree to disseminate multicast packets. This makes the paths between sources and members are no longer the shortest ones. As a result, members in shared trees suffer from longer end-to-end delay when compared to source-specific trees. Existing protocols (e.g. DVMRP [Waitzman, Partridge, and Deering 1988], MOSPF [Moy 1994], PIM-DM [Estrin *et al.* 1995], CBT [Ballardie 1997; Ballardie, Francis, and Crowcroft 1993], PIM-SM [Deering *et al.* 1996; Estrin *et al.* 1998], OCBT [Shields 1996; Shields and Garcia-Luna-Aceves 1997], and MIP [Parsa and Garcia-Luna-Aceves 1997]) are based on either source-specific trees or shared trees for multicast routing. We call the protocols based on source-specific trees as *source-specific tree approach* whereas those protocols based on shared trees as *shared tree approach* in this thesis.

1.1 The Problems

Regardless of whether the protocol considered is based on source-specific trees or shared trees, additional forwarding entries have to be maintained at on-tree routers so as to facilitate multicast routing. Multicast routing protocols based on source-specific trees (e.g. DVMRP, MOSPF, and PIM-DM) require a multicast forwarding entry for each source in a multicast group to be maintained at each on-tree router in these source-specific trees. Given a multicast group in which there are s sources, s source-specific trees will be built using these protocols. It is for this reason that the number of forwarding entries maintained at multicast routers is in proportion to the number of multicast groups and the number of sources in each multicast group. Let us suppose that there are g multicast groups. The number of forwarding entries maintained at multicast routers is, therefore, in proportion to $s \cdot g$. In other words, the overhead of source-specific tree approach is given by $O(s \cdot g)$. As a result, a large amount of forwarding entries is maintained at

multicast routers when there are a lot of sources in a multicast group or there are a lot of multicast groups.

On the other hand, multicast routing protocols based on shared trees (e.g. CBT, PIM-SM, OCBT, and MIP) require only a single forwarding entry per multicast group at each on-tree router because the same shared tree is used for all the sources to disseminate multicast packets. Consequently, the number of forwarding entries maintained at multicast routers is the same no matter how many sources there are in a multicast group. Nevertheless, more and more shared trees are built when the number of multicast groups increases. Given g multicast groups, g shared trees are built. The number of forwarding entries maintained at multicast routers is, therefore, in proportion to the number of multicast groups. In other words, the overhead of the shared tree approach is provided by $O(g)$. As a result, a large amount of forwarding entries is maintained at multicast routers when there are many multicast groups.

In general, the source-specific tree approach aims at achieving efficient end-to-end delay whereas the shared tree approach aims at achieving good scalability to the number of sources in a multicast group. Regardless of whether an existing multicast routing protocol considered is the source-specific or shared tree approach, it is developed to achieve either good scalability to the number of sources in a multicast group or efficient end-to-end delay but not both of them. This makes existing multicast routing protocols unable to support the Internet applications, which require efficient distribution of information over many sources and members.

The Internet applications of such kind include updating all copies of replicated databases by different hosts and disseminating and collecting intermediate results to and from a set of processors to support a distributed computation. The former application usually involves a few replicated databases but a lot of hosts widely distributed in different geographical regions. The updates on the databases should be as fast as possible in order to maximize the throughput of the application. It is for this reason that efficient delivery of information is critical for replicated databases. On the other hand, the latter application involves large amount of processors so as to support complicated computation in a distributed manner. The success of distributed computation relies on how fast the results of the calculation can be obtained. As a result, efficient dissemination and collection of information is one of the critical success factors for distributed computation. These applications and many others require both of good scalability and efficient end-to-end delay at the same time.

Furthermore, existing multicast routing protocols are inadequate to scale well when there are a large number of multicast groups no matter they are based on source-specific trees or shared trees. In fact, existing protocols based on shared trees, which are superior to those based on source-specific trees in terms of scalability, are developed to provide good scalability to the number of sources in a multicast group only. The overhead of the shared tree approach is $O(g)$ and they cannot scale well to the number of multicast groups. Nevertheless, it is possible that there are

a large number of multicast groups in large-scale the Internet applications. It is, therefore, critical that this problem is tackled in an effective manner.

In addition to the scalability problem to the number multicast groups and the number of sources in a multicasts group, the fact that multicast groups are sparsely represented in the Internet also introduces serious problems to those multicast routing protocols based on source-specific trees. To build multicast distribution trees, existing multicast routing protocols based on source-specific trees employ link-state or distance-vector techniques. Regardless of these protocols are link-state multicast routing protocols (e.g. MOSPF) or distance-vector multicast routing protocols (e.g. DVMRP), they are intended for use within regions where a multicast group is widely represented or bandwidth is universally plentiful. The occasional broadcasting nature of link-state protocols and the data driven nature of distance-vector protocols prevent themselves from suitable to wide-area networks where many multicast groups are sparsely represented. Unfortunately, some or all sources and members are, in general, sparsely represented in the Internet. The sparse nature of the Internet imposes serious problems to existing protocols. It is for this reason that the occasional broadcasting and the data driven issues have to be dealt with effectively.

In this thesis, we propose a multicast routing protocol called Adaptive Source and CORe Based MulticasT (ASCORT) and a technique for reducing forwarding states on top of any other multicast routing protocol called Tree Switching Protocol (TSP). ASCORT is based on neither source-specific trees nor shared trees. This protocol is able to achieve good scalability to the number of sources in a multicast group and efficient end-to-end delay at the same time. Unlike existing multicast routing protocols based on source-specific trees such as MOSPF and DVMRP, our protocol does not require occasional broadcasting and periodic flooding to maintain multicast distribution trees. TSP is developed to reduce multicast forwarding states required for a forest of source-specific trees or a forest of shared trees. It is able to achieve good scalability to the number of multicast groups. Furthermore, our protocols perform in a distributed manner and hence they do not have any single-point bottleneck. During the course of multicast tree construction, members in the multicast group are still able to receive multicast packets from the sources. In other words, there is no packet loss throughout the course of multicast tree construction. Together with the property that our protocols are loop-free, they can be effectively implemented in the real world.

1.2 Outline of Thesis

The rest of this thesis is organized as follows. In Chapter 2, we give a survey of related work in multicast routing. Specifically, we discuss the problem of providing good scalability to the number of sources in a multicast group and efficient end-to-end delay at the same time. We also discuss why existing multicast routing protocols are unable to achieve both of good scalability to the number of sources in a multicast group and efficient end-to-end delay. In this same chapter,

we also describe the scalability problem to the number of multicast groups and present how existing protocols are inadequate to deal with this problem.

In Chapter 3, we provide the details of ASCORT, a novel protocol for multicast routing. This protocol is able to achieve good scalability to the number of sources in a multicast group and efficient end-to-end delay at the same time. Existing multicast routing protocols are developed to provide either good scalability to the number of sources in a multicast group or efficient end-to-end delay but not both of them. This makes them unable to support the Internet applications which require efficient distribution of packets over many sources and members. The ability to provide good scalability to the number of sources in a multicast group and efficient end-to-end delay allows our protocol to be applicable in many real-world applications.

In Chapter 4, we describe TSP, which is capable of scaling to the number of multicast groups. Existing protocols only deal with the scalability problem to the number of sources in a multicast group. Nevertheless, the number of forwarding entries maintained at multicast routers depends not only on the number of sources in a multicast group but also on the number of multicast groups. To improve the scalability of existing multicast routing protocols, our protocol is able to scale to the number of multicast groups. This feature allows our protocol to be applicable to many large-scale Internet applications.

We conclude our study in Chapter 5. The major work carried out in our study is summarized. We also discuss some possible extensions to our works in this chapter.

Chapter 2

Survey of Related Work

In this chapter, we present the functionality a protocol is required to provide for supporting multicast routing in an effective manner. This chapter also gives a survey how multicasting capability is provided in existing routing protocols. These protocols, in general, construct a multicast distribution tree for efficient delivery of data. In this chapter, we describe different approaches for the construction of multicast distribution trees, namely, *Steiner trees*, *source-specific trees*, *shared trees*, hybrids of source-specific and shared trees, and hierarchical trees. Some of the protocols based on these approaches have limited scalability within a single domain whereas some of them are able to scale to the entire Internet. The former is known as *intra-domain multicast routing* and the latter is called *inter-domain multicast routing*. In this same chapter, we describe how existing multicast routing protocols can be used in intra-domain and inter-domain levels. Furthermore, we present how existing multicast routing protocols tackle (i) the problem in providing multicast routing for real-time applications; (ii) the problem in achieving good scalability to the number of sources in a multicast group and efficient packet distribution at the same time; and (iii) the problem in scaling to the number of multicast groups.

2.1 Requirements to Support Multicast Routing

2.1.1 Scalability

For a multicast routing protocol to be effectively applied at the intra-domain or the inter-domain level, it should scale well to the number of multicast groups and the number of sources in a multicast group. Since there will be many multicast groups and many sources in a multicast group in large-scale Internet applications, the method to deal with the problem of multicast forwarding state explosion is crucial to the success of the multicast routing protocol. Furthermore, the number of forwarding states for a multicast group should be minimized when there is no receiver or sender in that multicast group.

2.1.2 Efficient Data Distribution

In addition to good scalability, an effective multicast routing protocol should provide efficient distribution of data. In order to support low-delay data distribution, the path lengths between

sources and members in a multicast distribution tree should not be significant longer than the shortest paths.

2.1.3 Better Utilization of Network Bandwidth

To better utilize the network bandwidth, a routing protocol should spread multicast traffic across many routes instead of concentrating the traffic on only some routes when multiple sources send data simultaneously.

2.1.4 Independence of Other Routing Protocols

Multicast routing protocols should be independent of other routing protocols. Specifically, an intra-domain multicast routing protocol should be independent of the underlying unicast routing protocol. The multicast routing protocol can make use of the unicast routing tables but should not be dependent on the method for computing these tables. This allows the multicast routing protocol to rely on existing unicast routing functionality to adapt to topology changes in a domain and at the same time be independent of the particular protocol employed in that domain.

On the other hand, multicast routing protocols deployed at the inter-domain level should be independent of intra-domain multicast routing protocols. This allows each domain to choose a routing protocol, which is best suited for its need, without affecting any other domain. The independence of intra-domain multicast routing protocols also allows a domain to upgrade to a newer version of a protocol while minimizing the effects on other domains.

2.1.5 Robustness

Owing to the dynamic nature of the underlying networks in the Internet, a multicast routing protocol should be able to gracefully adapt to routing changes in order to provide a robust, multicasting environment to support multicasting. The multicast routing protocol should provide a mechanism to find an alternative route when the existing one becomes unreachable in the presence of link/node failures. Moreover, the multicast routing protocol should also avoid a single point of failure.

2.1.6 Minimal Control Overhead

Due to the overhead of additional control traffic and the potential loss of control messages, there is a tradeoff between optimal multicast distribution trees and protocol overhead. Such tradeoff is especially important in inter-domain multicast routing because of the scare bandwidth and the involvement of different domains. It has been suggested in [Kumar *et al.* 1998] that the reduction in protocol overhead is more important than the maintenance of optimal multicast distribution trees.

2.1.7 Routing Policy

For an inter-domain multicast routing protocol to be widely deployed in the Internet, it is important that the protocol has a policy model to control the flow of multicast traffic. The communication of two domains may depend on the quality of paths to a third-party domain in multicast routing protocols based on shared trees. It happens when the root of the shared tree lies in a third-party domain and the protocol requires multicast packets go via the root before reaching group members along the shared tree. Such dependency is not preferable because of administrative reasons. Instead of relying on the third-party domain, it is more desirable that the root of the shared tree is located in a domain that has either members or sources in the multicast group. It is important to note that this requirement is applicable to inter-domain multicast routing protocols only and is inapplicable to intra-domain multicast routing protocols.

2.2 Steiner Trees

Steiner trees (e.g. R-DGA [Aharoni and Cohen 1997], [Jia and Wang 1997], [Jia 1998]) aim at optimizing the use of network resources such as the use of network bandwidth for the multicast distribution trees. Given an undirected distance graph $G = (V, E, d)$ and a set S where V is the set of vertices in G, E is the set of edges in G, d is a distance function which maps E into the set of nonnegative numbers and $S \subseteq V$ is a subset of the vertices of V, the Steiner tree problem is to find a tree of G that *spans* S with minimum cost where the cost of a tree is the sum of the costs of its edges [Goemans and Myung 1993; Kou, Markowsky, and Berman 1981].

Unfortunately, it has been shown in [Karp 1972] that the problem of finding a minimal Steiner tree for any given G and S is NP-complete. This makes the computation of such trees to be very difficult. Even though many heuristic algorithms (e.g. R-DGA and [Kou, Markowsky, and Berman 1981]) have been developed for the Steiner tree problem, they do not guarantee to produce the optimal tree. Furthermore, a Steiner tree is very unstable as its form varies with the memberships of the multicast group [Huitema 1995]. Taking the dynamic nature of multicast group membership and the computation complexity of Steiner trees into consideration, the Steiner tree formulation is unable to provide an effective method for building multicast distribution trees. It is for this reason that Steiner trees are seldom used in existing intra-domain and inter-domain multicast routing protocols.

2.3 Source-Specific Trees

Some of existing multicast routing protocols (e.g DVMRP [Waitzman, Partridge, and Deering 1988], MOSPF [Moy 1994], and PIM-DM [Estrin *et al.* 1995]) are based on *shortest path trees* or *source-specific trees*. Since shortest paths between every pair of sources and members are available, source-specific trees are capable of providing optimal end-to-end delay. These protocols

such as *link-state* protocols (e.g. MOSPF) and *distance-vector* protocols (e.g. DVMRP) were intended for use within regions where a multicast group is widely represented or bandwidth is universally plentiful.

In order to construct source-specific trees using the link-state approach, changes of group memberships on a subnet are detected by one of the multicast routers attached to the subnet. This multicast router then broadcasts the group membership information to all other multicast routers in the same routing domain. Through the link-state unicast routing protocol such as OSPF [Moy 1998], all the multicast routers maintain an up-to-date image of the topology of the routing domain. Using the topology and the group membership information, a router performs shortest path computations for all the sources in the multicast group so as to construct source-specific trees from all the sources to all the members in that multicast group. Although link-state multicast routing protocols such as MOSPF can be very powerful protocols, the computation of shortest paths itself is a potential for saturating the CPUs of even the most powerful multicast routers [Huitema 1995]. This limits the applicability of building source-specific trees using link-state approach on an Internet-wide basis.

On the other hand, the distance-vector approach requires each multicast router to forward the first data packet addressed to a specific multicast group it received out of all interfaces except the incoming interface when the multicast router has no knowledge about that multicast group. When a multicast router attached to a leaf subnet receives a packet addressed to a new multicast group, the router will send a *prune message* upstream toward the source of the packet if it finds no member present on its attached subnet. The prune messages truncate the tree branches not leading to group members and hence a source-specific tree with all leaf routers having members is resulted. Pruned branches will grow back after a timeout period. These branches will again be pruned if there is still no member and multicast packets are still being sent to the multicast group. This data-driven nature of distance-vector approach requires off-tree multicast routers to perform periodic truncated-broadcast of packets and process subsequent pruning of branches.

It is important to note that both link-state protocols (e.g. MOSPF) and distance-vector protocols (e.g. DVMRP) are dependent on specific unicast routing protocols. In particular, link-state multicast routing protocols such as MOSPF depends on link-state unicast routing protocols such as OSPF whereas distance-vector multicast routing protocols such as DVMRP depends on distance-vector unicast routing protocols such as RIP. Unlike these protocols, PIM-DM, which is independent of any unicast routing protocol, has been developed to construct source-specific trees for multicast routing. Similar to distance-vector protocols, PIM-DM floods data packets and uses explicit pruning to set up the multicast distribution tree. However, PIM-DM does not depend on a specific unicast routing protocol to compute its reverse path and hence it is protocol independent.

Multicast routing protocols based on source-specific trees such as DVMRP, MOSPF, and PIM-DM aim at minimizing the end-to-end delay from each source to each member in a multicast

group by using the shortest path to connect each pair of sources and members. As a result, they achieve optimal end-to-end delay. When most of subnets in the Internet have group members or the multicast group is widely represented, group membership information in link-state protocols or data packets in distance-vector protocols are required in most parts of the Internet. It is for this reason that the bandwidth, storage, and processing overhead of broadcasting group membership information or data packets is legitimate.

Unfortunately, group members and multicast groups tend to be distributed across the Internet in a sparse manner [Deering *et al.* 1996]. The broadcasting nature of link-state protocols and the data-driven nature of distance-vector protocols prevent themselves from efficiently supporting sparsely distributed group members and multicast groups, which are more common in the Internet. Even worse, each router in link-state multicast routing protocols such as MOSPF is required to maintain information about the topology of the whole network. This makes these protocols unable to scale well when there are many routers in the network. Unfortunately, there are a lot of routers in the Internet because of its popularization. It is for this reason that link-state multicast routing protocols such as MOSPF are not readily applicable to facilitate multicast routing in large-scale Internet applications.

Furthermore, multicast routing protocols based on source-specific trees require a multicast forwarding entry for each source to be maintained at each on-tree router in these source-specific trees no matter they are link-state protocols (e.g. MOSPF), distance-vector protocols (e.g. DVMRP), or protocol-independent protocols (e.g. PIM-DM). Given a multicast group in which there are s sources, s source-specific trees will be built using these protocols. It is for this reason that the number of forwarding entries maintained at on-tree routers is in proportion to the number of sources in a multicast group, that is, s. Let us further suppose that there are g multicast groups. The number of forwarding entries maintained at multicast routers is, therefore, in proportion to $s \cdot g$. Accordingly, the overhead at a multicast router is given by $O(s \cdot g)$. As a result, a large amount of forwarding entries are maintained at multicast routers when there are many sources in a multicast group or there are many multicast groups.

Generally speaking, multicast routing protocols based on source-specific trees are suitable to interactive applications where efficient delivery of multicast packets is critical. Nevertheless, due to the occasional broadcasting behavior, these protocols are not readily applicable to large-scale Internet applications where sources and members are distributed sparsely in the Internet. Furthermore, the scalability problem of multicast routing protocols based on source-specific trees prevents them from being effectively used when there are many multicast groups and/or many sources in each multicast group.

2.4 Shared Trees

In addition to the source-specific approach, many existing multicast routing protocols such as CBT [Ballardie, 1997; Ballardie, Francis, and Crowcroft 1993], OCBT [Shields 1996; Shields and Garcia-Luna-Aceves 1997], PIM-SM [Deering *et al.* 1996; Estrin *et al.* 1997], QoSMIC [Banerjea, Faloutsos, and Rankaj 1998] and MIP [Parsa and Garcia-Luna-Aceves 1997] are based on shared trees. These protocols are suitable to the Internet applications where group members and multicast groups are sparsely distributed. A region is defined to be *sparse* if any one of the following conditions hold [Paul 1998].

1. The number of networks/domains with members is significantly smaller than the total number of networks/domains in a region.

2. Group members are widely distributed.

3. The overhead of flooding all the networks with data packets followed by pruning networks with no members in them is significantly high.

If none of the above conditions is satisfied, we consider the region to be *dense*.

The shared tree approach constructs a multicast distribution tree for each multicast group for any number of sources. The multicast distribution tree constructed by the shared tree approach is a shortest path tree rooted at a pre-selected router called *core* router connecting all members for a particular multicast group. A core router is a special router and is not necessarily has a source or a member in its LAN.

To construct a shared tree using PIM-SM, each designated router which has members for a multicast group in its LAN sends a *join* message toward a *rendezvous point* (RP). On receipt of a *join* message, a router creates a multicast forwarding entry for the multicast group. As a result, a shared tree rooted at the RP is constructed and each router on the tree maintains a multicast forwarding entry for the multicast group.

Unlike PIM-SM, CBT constructs a shared tree by requiring each designated router to send a *connection request* toward the core router for a multicast group in which it has members in its LAN. It also requires an on-tree router of the shared tree or the core router to return an *acknowledgement* to the initiated router upon receiving the connection request. The returning acknowledgement traverses along the reverse path of the connection request back to the initiated router. On receipt of an acknowledgement, a router creates a multicast forwarding entry for the multicast group. As a result, a shared tree rooted at the core router is built and each router on the tree maintains a multicast forwarding entry for the multicast group.

Recently, it has been shown in [Shields 1996; Shields and Garcia-Luna-Aceves 1997] that loops might be formed in the shared trees built by CBT. In order to overcome this problem, a variant of CBT, called OCBT, has been proposed in [Shields 1996; Shields and Garcia-Luna-Aceves 1997]. To construct such shared trees, multicast routers are organized in a hierarchy of *logical levels*. The lower-level core routers are allowed to join to higher-level core router only. This makes the root nodes of lower-level trees become the leaf nodes of the higher-level tree. Group members initiate the construction of multicast distribution trees by sending *join requests* toward the core routers. When the join requests reach the core routers or on-tree routers, *join acknowledgements* are sent back to the initiated member. The join acknowledgements are marked with the levels of the core routers they are intended for or on-tree routers reached. It is important to note that the levels of core routers are fixed whereas the levels of on-tree routers are variable and set by the returned join acknowledgements.

The multicast forwarding entry set up by a shared tree approach can either be "hard state" or "soft state." In particular, PIM-SM employs soft state mechanism whereas CBT employs hard state mechanism. If hard state mechanism is used, the multicast forwarding entry is removed explicitly. Unlike hard state mechanism, periodic refresh messages have to be sent to maintain the entry and the entry is timed-out if it is not refreshed within a certain interval in soft state mechanism. Both hard state and soft state mechanisms have their disadvantages. In soft state mechanism, periodic refresh messages are generated even in a stable network environment. On the other hand, a large amount of control messages are generated frequently if the network changes frequently if hard state mechanism is employed.

Let us note that the construction and maintenance of a shared tree is initiated by group members and is achieved by requiring group members send control messages toward the core router. Unlike the shared tree approach, the construction and maintenance of a multicast distribution tree is initiated by sources and is achieved by broadcasting multicast packet in source specific approach. It is this "receiver-initiated" nature of the shared tree approach that eliminates the occasional broadcasting behavior of the source-specific approach.

Regardless of a multicast routing protocol considered employs hard-state or soft-state mechanism, a source sends multicast packets toward the core router and the core router will disseminate the packets to the members on the shared tree. As a consequence, the overhead at a multicast router is $O(g)$ where g is the number of multicast groups. This gives multicast routing protocols based on shared trees a superior scalability when compared to multicast routing protocols based on source-specific trees.

Multicast routing protocols based on shared trees such as CBT, OCBT, PIM-SM, QoSMIC and MIP aim at tackling the scalability problem to the number of sources in a multicast group. Nevertheless, there are a large number of multicast groups in large-scale Internet applications. The multicast routers are therefore required to maintain a huge amount of forwarding state information.

This makes these protocols suffer from forwarding state explosion in large-scale Internet applications where there are many multicast groups.

Furthermore, shared trees incur longer end-to-end delay in the delivery of multicast packets because the packets no longer travel along the shortest path to each member in a multicast group. It is for this reason that these protocols are unable to provide optimal end-to-end delay. This makes multicast routing protocols based on shared trees inadequate to support the Internet applications which are highly delay sensitive or data intensive.

In addition to the inefficiency in packet delivery, multicast routing protocols based on shared trees tend to concentrate multicast traffics on certain links of the Internet since all the sources in a multicast group share the same multicast distribution tree. As a result, these protocols experience the problem in traffic concentration around the core router, which would easily become bottlenecks of multicast traffics [Wei and Estrin 1994].

It has also been shown that improper selection of the core router will result in poor performance of multicast routing protocols based on shared trees in terms of end-to-end delay and usage of bandwidth [Donahoo, Calvert, and Zegura 1997]. It is for this reason that additional core selection techniques such as [Calvert, Zegura, and Donahoo 1995] and [Donahoo, Calvert, and Zegura 1997] may be required in these protocols so as to achieve good performance.

2.4.1 Center Selection Protocols

In center (core) selection protocols described in [Doar and Leslie 1993; Donahoo, Calvert, and Zegura 1997; Wei and Estrin 1993; Shukla, Boyer, and Klinker 1994; Thaler and Ravishankar 1996; Voigt, Barton, and Shukia 1995], a center will be selected from a set of candidate routers for a multicast group. The set of candidate routers can be (i) all the routers in the network, (ii) all the first-hop routers for the multicast group in the network, (iii) all the member routers for the multicast group in the network, or (iv) all the first-hop routers and member routers for the multicast group in the network. The center will be (i) selected from the candidate routers randomly, (ii) the topological center of a set of candidate routers, or (iii) the candidate router with minimal cost.

The performance of the shared tree rooted at the selected center is highly dependent on the size of candidate router set. In general, the larger the set of candidate routers, the better the performance of the shared tree will be resulted. In addition to the size of candidate router set, it has also been shown in [Doar and Leslie 1993; Donahoo, Calvert, and Zegura 1997; Wei and Estrin 1993; Shukla, Boyer, and Klinker 1994; Thaler and Ravishankar 1996; Voigt, Barton, and Shukia 1995] that the shared tree rooted at the candidate router with minimal cost always produces the best performance among all the shared tree rooted at the center which is either randomly selected from the candidate routers or the topological center of the candidate routers.

It is important to note that the topological information of the network and the lists of sources and members are distributed across all the routers and hence no single router has complete information. A router has to exchange the topological information of the network and the lists of sources and members among neighboring routers regardless of how the center is selected. This introduces a large amount of control messages flowing in the network. Furthermore, the topological information of the network grows with the number of routers in the network whereas the lists of sources and members grow with the number of sources and members. The dynamic nature of group membership even makes the situation more complicated. Since existing sources/members will leave and new sources/members will join, the center selection protocols have to be executed again to find another center to adapt to the changes. As a result, large amount of control messages will flow in the network periodically in order to find a new center. In addition, migration of center will introduce the problem of packet loss.

2.5 Combination of Source-Specific and Shared Trees

Observing the limitations of source-specific trees and core-based trees, recent efforts have been made to combine these two approaches so that their advantages can be shared and those disadvantages can be eliminated. The synergy of source-specific and shared trees provides a possibility for a member to switch from a shared tree to a source-specific tree. This provides the flexibility for group members to tradeoff scalability to efficient packet distribution.

Let us note that PIM-SM, which builds shared trees, allows group members to receive packets from specific sources via source-specific routes. A member will send a *join message* to a specific source when it wants to receive packets from the source via a source-specific route. This results in the construction of a source-specific route connecting the source to the member. After establishment of the new source-specific route, the source will then make use of that route to deliver multicast packets to the member in the future. It is for this reason that PIM-SM is able to construct a combination of source-specific and shared trees.

In addition to PIM-SM, QoSMIC is an example of such hybrid approach. QoSMIC constructs a shared tree at first and then allows members to receive packets from specific sources via shortest paths or source-specific routes. When a host wants to join a multicast group, its designated router becomes a member router and will perform the *local search* and the *multicast tree search* at the same time. During the course of local search, the member router searches its neighboring on-tree routers by flooding a *probe message* in its neighborhood. Every on-tree router that receives the probe message responds with an *advertisement message* unicast to the member. On the other hand, the member router contacts the *Manager router* of the group in the multicast tree search. The Manager router provides some candidates for the member to join. Those candidates advertise themselves to the member with a unicast advertisement message. As a result,

alternative paths for each connection are provided to the member in order to choose the best path according to the requirements of the application.

QoSMIC is able to provide several paths for new members to select so that the best path according to the requirements of the application can be chosen. Furthermore, QoSMIC can reduce the loading of the core router by separating the management and data transfer. Nevertheless, the overhead introduced by the local search and multicast tree search makes the use of QoSMIC [Banerjea, Faloutsos, and Rankaj 1998] in large-scale Internet applications become inefficient.

It is important to note that multicast routing protocols based on a combination of source-specific and shared trees construct shared trees and then allow source-specific routes to be established. When no source-specific routes are established, the scalability of these protocols is the same as that of the shared tree approach. In other words, they scale well to the number of sources in a multicast group. Nevertheless, if many source-specific routes are set up, many multicast forwarding entries are maintained at the routers en route. The worst case is that source-specific routes are established for every pair of sources and member. The shared tree is then transformed to a forest of source-specific trees. In that case, these protocols suffer from the scalability problem just as source-specific tree approach. The scalability of hybrid approach can be considered to be lying in the continuum such that one of its ends is the scalability of the shared tree approach and the other end is the scalability of source-specific tree approach.

Similarly, when group members receive multicast packets on the shared tree, the efficiency in packet distribution of these multicast routing protocols is the same as that of the shared tree approach. However, the members are also allowed to receive multicast packets via source-specific routes from specific sources. This provides a mechanism to tradeoff scalability to efficient packet delivery. When every member receive multicast packets from all sources via source-specific routes, the efficiency in packet delivery of these protocols is the same as that of the source-specific approach. The efficiency in packet delivery of hybrid approach can be considered to be lying in the continuum such that one of its ends is the efficiency in packet distribution of the shared tree approach and the other end is the efficiency in packet delivery of source-specific tree approach.

2.6 Hierarchical Trees

Recently, construction of multicast distribution trees in a hierarchical manner has been proposed in the literature. One of these approaches called HDVMRP [Thyagarajan and Deering 1995] involves partitioning a network into non-overlapping *regions*. The network is organized into a two-level hierarchy – the top-level consisting of regions and the lower level consisting of subnets within the regions. Each region has a unique *region identifier*. These regions are interconnected by *boundary routers* which run a Level 2 (L2) multicast routing protocol whereas the routers internal to a region run a Level 1 (L1) multicast routing protocol. Any multicast routing protocol

can be used within a region for L1 multicast routing. However, only DVMRP is used as the L2 multicast routing protocol for inter-region multicasting. This hierarchical approach allows the boundary routers to exchange information about only the regions as opposed of about every subnet in every region and thereby reduces the amount of information exchanged between the routers and also reduces the number of entries in the routing tables [Paul 1998].

In HDVMRP, multicast packets are encapsulated for a boundary router of one region to send to boundary routers of another region. These encapsulated packets appear as regular multicast packets to a region. It is important to note that hosts, which are receivers of multicast traffic from external sources, could have the same packet traverse a link in the encapsulated form as well as the original form. To avoid this problem, *router decapsulation* and *host decapsulation* can be used. Unfortunately, both of these methods introduce additional overhead in examining each transit encapsulated packet. Furthermore, the encapsulation/decapsulation mechanism prevents HDVMRP from supporting QoS which has been considered as one of the essential features on the Internet in the future.

In addition to HDVMRP, a hierarchical version of PIM called HPIM has been proposed in [Handley, Crowcroft, and Wakeman 1997]. In HPIM, *rendezvous points* (RPs) are arranged in a hierarchy such that each *candidate* RP (C-RP) has a level number associated with it. The higher the level, the large the coverage area per C-RP. Designated routers connecting to the sources and members are called level 0 RPs. The C-RPs at level n announce their availability by multicasting with a fixed TTL. The C-RPs at level $n - 1$ compute their distance from level n C-RPs and pass that information to level $n - 2$ C-RPs while also advertizing their own availability. This process continues until the C-RPs at level 0 have complete information about C-RPs at all the other levels.

A host informs its designated router (i.e. level 0 RP) using IGMP when it wants to join a multicast group. The designated router then sends a *join message* toward its level 1 RP, which then sends a *join-acknowledgement message* back to the level 0 RP. The level 1 RP repeats the same process with the level 2 C-RP list and finds the corresponding level 2 RP toward which it sends the join message. Upon receiving the join message, the level 2 RP sends a join-acknowledge message to the level 1 RP. This process continues until an RP is reached at the maximum level for the multicast group.

When a host starts sending packet to a multicast group, its designated router encapsulates the packet in a *register message* and sends it to the next hop router toward the corresponding level 1 RP. On the receipt of the register message, the level 1 RP sends a *register-acknowledge message* back to the level 0 RP. The level 1 RP also decapsulates the register message and checks if it is a part of the shared tree. If it does, the register message is not forwarded any further. Otherwise, the register message is re-encapsulated and forwarded to the next-hop router toward the level 2 RP. The level 2 RP repeats the same process until the register message reaches an RP which is already a part of the shared tree.

The hierarchical arrangement of RPs allows HPIM to detect loops and decouple the control flow from the data flow. Even if control packets follow a sub-optimal route, data packets follow an improved route as is apparent from the source-specific routes and the pruning of RPs [Paul 1998]. A weakness of this protocol is that extensive knowledge of the network topology and the receiver set is required to form the hierarchy of RPs [Shields 1996; Shields and Garcia-Luna-Aceves 1997]. This limits the applicability of this protocol to the Internet-wide applications. Moreover, HPIM is homogeneous in such a way that each domain has to use PIM as intra-domain multicast routing protocol. This further limits the applicability of HPIM in Internet-wide basis because it is very difficult to enforce every domain to deploy PIM for multicast routing. Let us further note that HPIM requires pre-specified RPs. The resulting multicast distribution trees are therefore unable to adapt to the locations of sources and members. It is for this reason that efficient packet delivery can hardly be achieved.

2.7 Border Gateway Multicast Protocol

Multicast routing protocols such as DVMRP, MOSPF, PIM-DM, CBT, PIM-SM, and MIP are only appropriate for multicast routing in a single domain. Specifically, the flooding and pruning mechanism employed in distance-vector multicast routing protocols (e.g. DVMRP) is not readily applicable to wide-area multicast with sparsely distributed members. Link-state multicast routing protocols (e.g. MOSPF) has the scalability limitations of link-state unicast routing protocol (e.g. OSFP) which does not scale beyond 200 nodes, a number far less than the number of autonomous systems or domains [Paul 1998]. Furthermore, both distance-vector and link-state multicast routing protocols are dependent on specific unicast routing protocols. This prevents them from being deployed in the inter-domain level because it is very difficult to enforce all domains to use corresponding unicast routing protocols. Even though PIM-DM is unicast protocol independent, the fact that it employs flooding and pruning mechanism prohibits itself from being used for inter-domain multicast routing.

In addition to multicast routing protocols based on source-specific trees (e.g. DVMRP, MOSPF, and PIM-DM), multicast routing protocols based on shared trees such as CBT, PIM-SM, and MIP are also inappropriate for inter-domain multicast routing. To use these protocols (e.g. CBT, PIM-SM, and MIP) in the inter-domain level, it is possible that the core routers are located in a domain with poor network connectivity. This results in bad performance for the entire multicast group. Furthermore, it is also possible that the core routers are located in a domain far form any member in the multicast group. This disallows multicast routing protocols based on shared trees such as CBT, PIM-SM, and MIP to provide efficient packet distribution.

Multicast routing protocols which construct a combination of source-specific and shared trees such as PIM-SM and QoSMIC have to build a shared tree at the very beginning. The construction of the shared tree makes them suffer from the same limitations as those protocols

based on shared trees. Hierarchical multicast routing protocols (e.g. HDVMRP, HPIM, and OCBT) are developed to address the scalability problem in wide-area multicast. Unfortunately, they are unable to observe the routing policy which has been considered as one of the most important critera in inter-domain multicast routing [Kumar *et al.* 1998; Thaler, Estring, and Meyer 1998].

Owing to the limitations of existing multicast routing protocol, Border Gateway Multicast Protocol (BGMP) [Kumar *et al.* 1998; Thaler, Estring, and Meyer 1998] is developed to address the issue of inter-domain multicast routing. BGMP is able to interoperate with other intra-domain multicast routing protocols such as DVMRP, MOSPF, PIM-DM, CBT, PIM-SM, and MIP in the transit domains. In BGMP, MASC [Estrin *et al.* 1997; Kumar *et al.* 1998] is used to allocate multicast address prefix so that a *root domain* can be chosen. BGMP builds a *bi-directional shared tree* rooted at the root domain connecting *border routers*.

Specifically, if a receiver wants to join a multicast group, the border router of the receiver's domain generates a *group-specific join message*, which is forwarded across border routers until it reaches either the root domain or an existing branch of a BGMP multicast tree. All routers en route create a *group-specific bi-directional state* such that any multicast packet destined to the group is forwarded on all the interfaces of the BGMP multicast tree other than the incoming interface of the packet. A nice feature of BGMP is the provision for attaching *source-specific branches* as "short cuts" to avoid the otherwise long delay imposed by a shared tree.

2.8 Techniques for Tackling the Problems

In this section, we describe how existing multicast routing protocols can be used to tackle the problems we discussed in Section 1.1. Specifically, they are (i) providing efficient multicast routing for real-time applications in the Internet where group members are sparsely represented; (ii) achieving good scalability to the number of sources in a multicast group and efficient packet distribution at the same time; and (iii) scaling to multicast groups.

2.8.1 Multicast Routing for Real-Time Applications in the Internet

Multicast routing protocols based on source-specific trees (e.g. DVMRP, MOSPF, and PIM-DM) are able to provide shortest paths between each source and each member in a multicast group. It is for this reason that they are capable of providing efficient multicast packet delivery for real-time applications such as replicated databases and distributed computation. Existing protocols such as link-state multicast routing protocols (e.g. MOSPF) and distance-vector multicast routing protocols (e.g. DVMRP) are developed for interneworks where a multicast group is widely represented or bandwidth is universally plentiful. In order to build multicast trees, the link-state protocols involve occasional broadcasting whereas the distance-vector protocols make use of data-driven techniques.

Such techniques allow these protocols to provide multicast routing effectively in dense-mode internetworks.

Unfortunately, it is common that some or all group members are sparsely distributed in the Internet. The broadcasting nature of link-state protocols and the data-driven nature of distance-vector protocols make them not readily applicable to the Internet applications because only a small number of subnets are involved in a multicast group and network bandwidth is limited in the Internet. In other words, existing protocols based on source-specific trees no matter they build multicast distribution trees using link-state or distance-vector techniques are inadequate to support many large-scale Internet applications effectively.

2.8.2 Scaling to Number of Sources and Efficient Multicast Packet Delivery

Unlike source-specific tree approach (e.g. DVMRP, MOSPF, and PIM-DM), multicast routing protocols based on shared trees (e.g. CBT, PIM-SM, and MIP) aim at achieving good scalability to the number of sources in a multicast group. Specifically, the overhead at a multicast router in these protocols is $O(g)$ instead of $O(s \cdot g)$ as in multicast routing protocols based on source-specific trees where s is the number of sources in a multicast group and g is the number of multicast groups. This makes multicast routing protocols based on shared trees such as CBT, PIM-SM, and MIP independent of the number of sources in a multicast group and hence enables them to scale well to the number of sources in the multicast group.

Regardless of whether an existing multicast routing protocol considered is based on source-specific trees or shared trees, it is developed to achieve either efficient end-to-end delay or good scalability to the number of sources in a multicast group but not both of them. This makes existing multicast routing protocols such as DVMRP, MOSPF, PIM-DM, CBT, PIM-SM, and MIP unable to support those Internet applications such as replicated databases and distributed computation, which require efficient delivery of information over many sources and members.

In order to provide both of good scalability to the number of sources in a multicast group and efficient end-to-end delay at the same time, a multicast architecture called PIM is described in [Deering *et al.* 1996; Estrin *et al.* 1995, 1998]. PIM supports both source-specific and shared trees in the same architecture. In PIM, PIM-DM is used as the protocol to build source-specific trees whereas PIM-SM is used as the protocol to build shared tree. A PIM router supports both PIM-DM and PIM-SM. When the PIM router is used in a region where a multicast group is widely represented or bandwidth is universally plentiful, it can be configured to use PIM-DM as multicast routing protocol. On the other hand, when the PIM router is used in an environment where group members and multicast groups are distributed sparsely across a wide area, it can be configured to use PIM-SM for multicast routing.

It is important to note that a PIM router is configured to use either PIM-DM or PIM-SM. This implies that it is able to achieve either scalability or efficient packet distribution but not both

at the same time. Even though PIM-SM allows members to receive multicast packets from specified sources via source-specific routers, it represents a means to tradeoff scalability to efficient packet delivery rather than a means to achieve scalability and efficient packet delivery at the same time.

In addition to PIM-SM, multicast routing protocols which constructs a combination of source-specific and shared trees such as QoSMIC also build a shared tree at first and then allow members to receive multicast packets from specified sources via source-specific routes. Again, this should be considered as a means to tradeoff scalability to efficient packet distribution instead of a way to achieve both scalability and efficient packet delivery at the same time.

2.8.3 Scaling to Number of Multicast Groups

In the Internet multicast architecture, there is no "natural" limit to the number of concurrent multicast groups. Unlike unicast routing, multicast receivers are not topologically related. Moreover, the number of multicast groups grows combinatorially with the number of network nodes. If some significant fraction (say 50%) of multicast groups are simultaneously in use, some routers may need more than 0.5 GB memory to store the forwarding entries [Radoslavov, Estrin, and Govindan 1999]. This can seriously inhibit existing multicast routing protocols to be deployed in the Internet-wide applications. To support large-scale Internet applications, efficient strategies for scaling to the number of multicast groups are necessary.

Given g multicast groups and s sources in each multicast group, the overhead at multicast routers in protocols based on source-specific trees such as DVMRP, MOSPF, and PIM-DM are $O(s \cdot g)$ whereas the overhead at multicast routers in protocols based on shared trees such as CBT, PIM-SM, and MIP are $O(g)$. The scalability of a multicast routing protocol is therefore affected by the number of multicast groups no matter the protocol considered is based on source-specific trees or shared trees.

Existing multicast routing protocols (e.g. DVMRP, MOSPF, PIM-DM, CBT, PIM-SM, and MIP) will face the problem of multicast forwarding state explosion when the number of groups becomes large. Although the problem in scaling to the number of multicast groups is as important as the scalability problem to the number of sources in a multicast group, it has not been widely studied in the literature [Aharoni and Cohen 1997; Ballardie 1997; Ballardie, Francis, and Crowcroft 1993; Banerjea, Faloutsos, and Pankaj 1998; Deering 1991; Deering *et al.* 1996; Estrin *et al.* 1995, 1997; Faloutsos, Banerjea, and Pankaj 1998; Floyd *et al.* 1996; Garcia-Luna-Aceves and Madruga 1999; Kompella, Pasquale, and Polyzos 1993; Moy 1994; Shields 1996; Shields and Garcia-Luna-Aceves 1997; Parsa and Garcia-Luna-Aceves 1995, 1997; Rouskas and Baldine 1997; Salama 1996; Tode *et al.* 1992; Waitzman, Partridge, and Deering 1988].

In general, state aggregation techniques can be used to tackle this scalability problem. A straightforward approach, called *non-leaky* aggregation strategy, is applicable to such problem.

This strategy preserves the semantics of multicast joins, distributing data only in the direction of receivers. It therefore does not tradeoff bandwidth for reduced table size. In this strategy, two adjacent group-specific forwarding entries are aggregated if and only if their incoming interfaces match and their outgoing interface lists match. This form of aggregation is also known as *strict* aggregation. However, there is little likelihood that many group prefixes satisfy these conditions. It is for this reason that the amount of state reduction is insignificant.

To overcome this limitation to certain extent, a variant of this strategy, called *pseudo-strict* aggregation, can be used. In this form of aggregation, two group prefixes are replaced by the longest covering prefix if their incoming interfaces match, their outgoing interface lists match, and there is no intervening forwarding table entry whose group prefix matches the longest covering prefix [Radoslavov, Estrin, and Govindan 1999]. Even thought pseudo-strict aggregation is expected to provide better compression of forwarding states than strict aggregation, the amount of state aggregation is still insignificant as reported in [Radoslavov, Estrin, and Govindan 1999].

Unlike non-leaky aggregation strategy, a *leaky* aggregation strategy has been proposed in [Radoslavov, Estrin, and Govindan 1999] to reduce multicast forwarding states. The leaky aggregation strategy relaxes the requirement in pseudo-strict aggregation that the outgoing interface lists of the entries must match [Radoslavov, Estrin, and Govindan 1999]. In leaky aggregation strategy, a multicast packet that matches the resulting forwarding entry will not only be forwarded on all interfaces on which *Join messages* have been received but also be forwarded on some other interfaces as well. It is for this reason that some subnets, which have not sent Join messages, receive the packet. The leaky aggregation strategy clearly wastes some of the bandwidth. This represents a tradeoff between the usage of bandwidth and the reduction in forwarding states.

In addition to leaky and non-leaky aggregation strategies, [Tian and Neufeld 1998] represents another attempt to address the problem of reducing the number of multicast forwarding states. Based on the observation that the multicast distribution tree of a group is likely to contain long and unbranched paths when the members of a group are sparsely located, DTM makes use of dynamically established tunnels between the start and end points of the unbranched paths to eliminate multicast forwarding states maintained at routers on these paths [Tian and Neufeld 1998]. The dynamic tunnels encapsulate multicast packets to bypass the unbranched nodes. The encapsulated packets are forwarded in a unicast manner between the end points of the dynamic tunnels. As a result, the unbranched nodes are not required to maintain any information about the multicast group. This allows the elimination of multicast forwarding state information maintained at unbranched nodes and DTM only requires the root node, branching nodes, and leave nodes of the original multicast distribution tree to maintain multicast forwarding entries.

The elimination of the forwarding states on the unbranched nodes is able to greatly reduce the overall forwarding state improvement and hence achieve good scalability to the number of

multicast groups [Tian and Neufeld 1998]. Unfortunately, there is additional encapsulation and decapsulation overhead per data packet. Furthermore, this approach is not beneficial if the fanout of the groups is more than one. The use of tunnels to disseminate packets on the multicast distribution tree also disallows resource reservations, which is considered as one of the fundamental components for accommodating multicast in the Internet [Banerjea, Faloutsos, and Pankaj 1998; Braden *et al.* 1997; Zhang *et al.* 1993]. In resource reservation protocols such as RSVP [Braden *et al.* 1997; Zhang *et al.* 1993], resources are reserved in each node along the data path. DTM makes such kind of reservation become infeasible because multicast packets are encapsulated in dynamic tunnels. The widespread demands on providing Quality of Service (QoS) applications in the Internet and the incapability of supporting QoS limit the applicability of DTM.

Chapter 3

Adaptive Source and Core Based Multicast: A New the Internet Multicast Routing Protocol

In this chapter, we propose a novel protocol called Adaptive Source and CORe Based MulticasT (ASCORT) for constructing multicast distribution trees. ASCORT provides an alternative to multicast routing protocols based on shared trees by achieving good scalability and efficient end-to-end delay at the same time. Specifically, ASCORT constructs a forest of reduced trees and a bi-directional tree without having any pre-specified core router. A reduced tree is rooted at the first-hop router of one of the sources connecting only a subset of group members whereas a bi-directional tree is rooted at the first-hop router of a source connecting the first-hop routers of all the other sources. The technique of building reduced trees is able to partition group members into different disjoint subsets. Using this technique, each multicast router is required to maintain one forwarding entry only. This technique also allows each group member to receive multicast packets via the shortest path from one of the sources. Furthermore, the use of bi-directional tree to connect sources together makes ASCORT become adaptive to the location of sources and it is for this reason that ASCORT is able to obtain efficient end-to-end delay. The simulation results show that ASCORT is able to achieve better scalability and more efficient end-to-end delay than PIM-SM, a well-known multicast routing protocol based on shared trees. Together with the fact that ASCORT is a distributed protocol, routing loop free, and very efficient, ASCORT is applicable to large-scale Internet applications in the real world.

3.1 Introduction

In existing multicast routing protocols, source-specific trees are built to achieve efficient packet distribution whereas shared trees are constructed to achieve good scalability to the number of sources in a multicast group. Recently, some techniques (e.g. PIM-SM and QoSMIC) are proposed to allow switching from a shared tree to source-specific routes so that members are able to receive multicast packets from specified sources via shortest paths. This provides flexibility for members to choose receiving packets on the shared tree or via source-specific routes. In the extreme case that all members require efficient packet delivery from all sources, a forest of source-specific trees are built. Good scalability can be accomplished if members choose to receive packets on the shared tree whereas efficient packet delivery can be achieved if members decide to

receive packets via source-specific routes. In other words, these techniques provide a means to tradeoff good scalability with efficient packet distribution and vice versa. Regardless of an existing multicast routing protocol is based on source-specific trees, shared trees, or combination of them, it is developed to achieve either efficient packet delivery or good scalability but not both of them at the same time.

Unlike these protocols, we propose a novel multicast routing protocol called Adaptive Source and CORe Based MulticasT (ASCORT) which is able to achieve good scalability and efficient packet distribution at the same time. To achieve this goal, we partition group members into different disjoint subsets. We then allow each member to connect to a source. This results in a forest of trees each of which is rooted at a source and connects a subset of members. We call such trees as *reduced trees*. ASCORT then connects the roots of these reduced trees in the form of a tree. The branches of the tree is bi-directional which allows multicast packets traverse from one on-tree router to another and vice versa. We call the tree as *bi-directional tree* and the root of the bi-directional tree as *core*. ASCORT can select a "good enough" core to provide efficient packet distribution without additional center selection protocols.

Furthermore, ASCORT is able to achieve good scalability and efficient packet distribution with minimal control overhead. ASCORT eliminates the necessity of having a pre-selected core/RP in CBT/PIM-SM. Moreover, ASCORT is capable of adapting to network changes and membership dynamics. The multicast distribution trees constructed by ASCORT are free of any loop. In addition, ASCORT is independent of any unicast routing protocol and is able to support Quality of Service (QoS), which is considered as one of the fundamental components for accommodating multicast in the Internet [Banerjea, Faloutsos, and Pankaj 1998; Braden *et al.* 1997; Zhang *et al.* 1993]. The details of these objectives are given in the following.

Scalability

Since there are many sources in a multicast group in large-scale Internet applications, the success of a multicast routing protocol depends on the ability to effectively deal with the problem of multicast forwarding state explosion.

Efficient Packet Distribution

The successes of many Internet applications depend on efficient delivery of data. A multicast routing protocol should therefore achieve efficient packet distribution in addition to good scalability.

Effective Method for Core Selection

In multicast routing protocols based on shared trees such as CBT and PIM-SM, a well-known or pre-selected core/RP is required to build multicast distribution trees. Furthermore, the selection of

such core/RP may require additional center selection protocols in order to achieve efficient delivery of packets. When a center selection protocol is used, a large amount of control messages are generated which consume bandwidth and may overload the network. To support multicast service effectively, a multicast routing protocol should be able to automatically select a "good enough" core router without using any center selection protocol.

Minimal Control Overhead

The bandwidth is limited in the Internet and hence control traffics used in a multicast routing protocol should be kept as small as possible. It has been suggested in [Kumar *et al.* 1998] that the reduction in protocol overhead is more important than the maintenance of optimal multicast distribution trees.

Adaptability of Network Dynamics

In the Internet, it is possible that some routers and some links in the network fail. An effective method should be employed in multicast routing protocols to adapt to network dynamics.

Adaptability of Group Dynamics

In multicast applications, it is possible that (i) a new source joins a multicast group; (ii) a new member joins a multicast group; (iii) a source in a multicast group becomes inactive; and (iv) a member in a multicast group leaves the group. A multicast routing protocol should provide an effective technique to handle these group dynamics.

Loop Freedom

Since applications using multicasting tend to be real-time and bandwidth intensive, loops in multicast routing duplicate looping packets for a large volume of packets. The multitude of duplicate packets consumes bandwidth and increases queuing delay along a subtree of the multicast distribution tree. To provide multicast service effectively, a multicast routing protocol should build multicast distribution trees, which are free of any loop, when no loop is formed in unicast routing.

Independence of Unicast Routing Protocols

A multicast routing protocol should be independent of the underlying unicast routing protocol. The multicast routing protocol can make use of the unicast routing tables but should not be dependent on the method for computing these tables. This allows the multicast routing protocol to rely on existing unicast routing functionality to adapt to topology changes in a domain and at the same time be independent of the particular protocol employed in that domain.

Support for Quality of Service

Quality of Service (QoS) has been considered as one of the essential features on the Internet in the future. Therefore, a multicast routing protocol should support QoS when deploys in the Internet.

The rest of this chapter is organized as follows. In Section 3.2, we provide an overview of ASCORT. Specifically, we discuss how ASCORT achieves the requirements described in this section. In Section 3.3, we present the detailed protocol of ASCORT. In particular, we describe (i) control messages; (ii) multicast forwarding entries; (iii) multicast forwarding algorithm; (iv) the construction of multicast distribution trees including reduced trees and bi-directional trees; (v) static maintenance of multicast distribution trees; (vi) adaptability to group dynamics; (vii) adaptability to network dynamics; and (viii) the use of timers. In Section 3.4, we examine the properties of ASCORT. To evaluate the performance of ASCORT, we perform simulation experiments and the results are reported in section 3.5. Finally, in Section 3.6, we conclude this chapter with a summary.

3.2 An Overview of ASCORT

In this section, we provide how ASCORT achieves the requirements described in Section 3.1.

Scalability

To achieve good scalability to the number of sources in a multicast group, ASCORT connects a member to one source only. We called such source as *selected source*. Instead of connecting the member to all the sources just as those multicast routing protocols based on source-specific trees, each member is connected by a tree branch rather than several tree branches and hence the number of on-tree routers is reduced. To determine the selected source, a member evaluates the numbers of hops along the shortest paths to all sources. The nearest one in terms of hop count is chosen as the selected source. This, in effect, minimizes the number of routers on the path. As a result, the number of forwarding entries maintained at on-tree routers can be reduced. By connecting members to their nearest sources, these members are partitioned into disjoint subsets. This prevents the reduced trees from overlapping with one another and hence eliminates the possibility that a router maintains more than one multicast forwarding entries for a multicast group. In other words, multicast routers lying on the reduced trees are required to maintain one forwarding entry only. This significantly reduces the number of entries in the multicast routing table maintained at each multicast router.

In addition to construct reduced trees, a bi-directional tree is also constructed to connect all the reduced trees together. A multicast forwarding entry is maintained in each router lying on the bi-directional tree. In case that a bi-directional tree overlaps with a reduced tree (e.g. a root of reduced tree always lying on a bi-directional tree), only one multicast forwarding entry is

maintained at the routers lying on the overlapping branches. As a result, only one multicast forwarding entry is maintained at the routers lying on the distribution tree constructed by ASCORT.

Efficient Packet Distribution

In ASCORT, a forest of reduced trees and a bi-directional tree are constructed. When a source sends a multicast packet, a copy is sent to its reduced tree and a copy is sent to the bi-directional tree. The multicast packet is traversed in the reduced tree to reach a subset of members. It is important to note that the average end-to-end delay on the reduced tree rooted at a source is even less than that on the source-specific tree rooted at that source connecting all group members because each member joins the reduced tree rooted at the nearest source.

In addition to the reduced trees, the multicast packet has also to traverse the bi-directional tree. Through the bi-directional tree, the multicast packet can reach other reduced trees and hence other group members. Similar to PIM-SM in which a multicast packet traverses a shared tree to reach all group members, a multicast packet traverses a bi-directional tree to reach all group members in ASCORT. The efficiency of packet distribution of a shared tree depends on the end-to-end delay of the tree. The end-to-end delay, in turn, highly depends on the location of a RP. Since a RP in PIM-SM is pre-selected without the knowledge about the location of sources and members, the end-to-end delay of a shared tree may be poor.

Similarly, the efficiency of packet distribution in a multicast distribution tree constructed by ASCORT heavily depends on the location of the core. Unlike PIM-SM, the core is selected from a set of first-hop routers in ASCORT. To show why the selection of the core from first-hop routers is able to achieve efficient packet distribution, we perform the following simulation experiments.

In the experiments, we evaluate the average end-to-end delay of a shared tree. We consider two different ways to select a RP of a shared tree: (i) the RP is selected from all the routers in the network randomly (All Nodes); and (ii) the RP is selected from a set of first-hop routers randomly (Sources). The RP is then connected to all members in the multicast group using a shared tree. In addition, the RP is connected to each source using a source specific path.

We generate a network consists of 100 routers. We set the average nodal degree to four and the cost of each link is assumed to be unity. We set the number of members to 20 and the number of sources increases from 2 to 10. Sources/members in a multicast group may be distributed in a random manner over the network. In other words, each node has the same probability to be a source/member. We call such distribution as *random distribution*. On the other hand, sources/members may be clustered together in some regions in the network. In other words, the node whose neighbors are sources/members has higher probability to be a source/member. We

call this distribution as *localized distribution*. To analyze the average end-to-end delay of the bi-directional tree thoroughly, we consider the following distributions of sources and members.

Scenario 1: random distribution of sources and members;
Scenario 2: random distribution of sources and localized distribution of members;
Scenario 3: localized distribution of sources but random distribution of members; and
Scenario 4: localized distribution of sources and members.

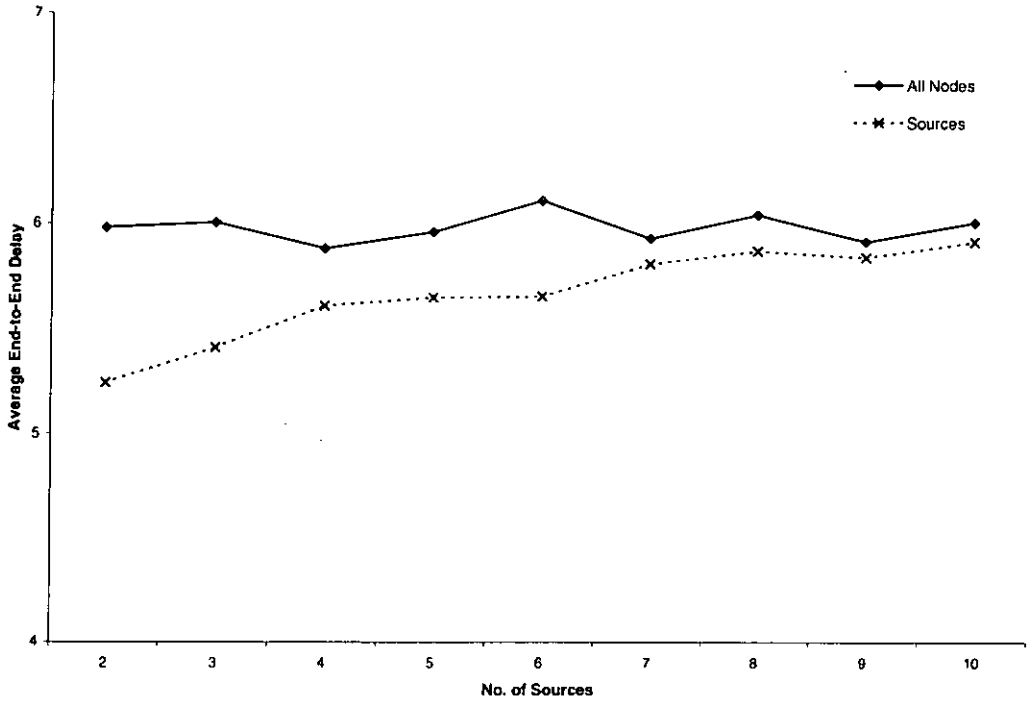
Let us note that each data point in our simulation results is an average over 100 independent simulation experiments. The results are shown in Fig. 3.1.

The results show that better average end-to-end delay can be obtained if the RP is selected from a set of first-hop routers rather than selected from all routers in the network randomly under all the four scenarios. In particular, the average end-to-end delay obtained by selecting the RP from a set of first-hop routers is significantly less than that obtained by selecting the RP in the network randomly in localized distribution of sources (Scenario 3 and 4). The selection of RP from first-hop routers allows it to better adapt to the location of sources and hence provides efficient packet distribution in the distribution tree.

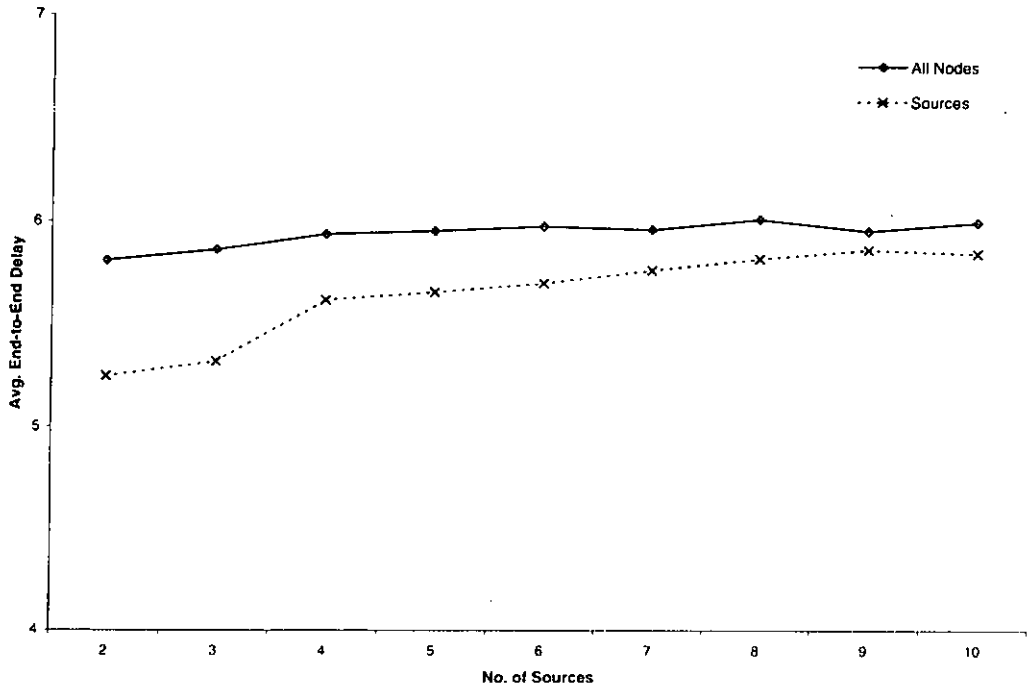
Based on the above observations, ASCORT selects the core from the first-hop routers in order to obtain better end-to-end delay in a bi-directional tree. It is for this reason that ASCORT is able to provide efficient packet distribution in the multicast distribution tree.

Effective Method for Core Selection

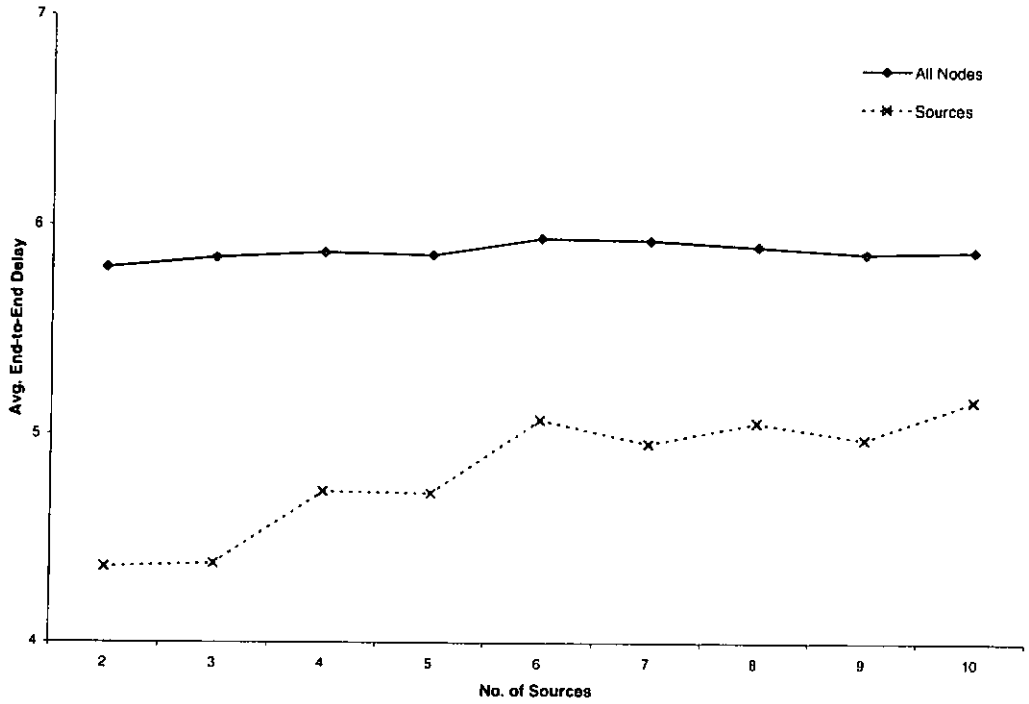
In ASCORT, a new source broadcasts its first multicast packet to all routers in the network. As a result, all routers have the knowledge about the source. Routers make use of this information to select the first host sending to a multicast group as the *base source* and the base source's first hop router as the core for that multicast group. This eliminates the necessity of a pre-selected core/RP in CBT and PIM-SM. Moreover, ASCORT does not generate any control message in the process of finding a "good enough" core. Furthermore, ASCORT is capable of selecting a core, which can adapt to the location and sources, without using any center selection protocol. Our experimental results above show that this selection method is able to provide efficient packet distribution.



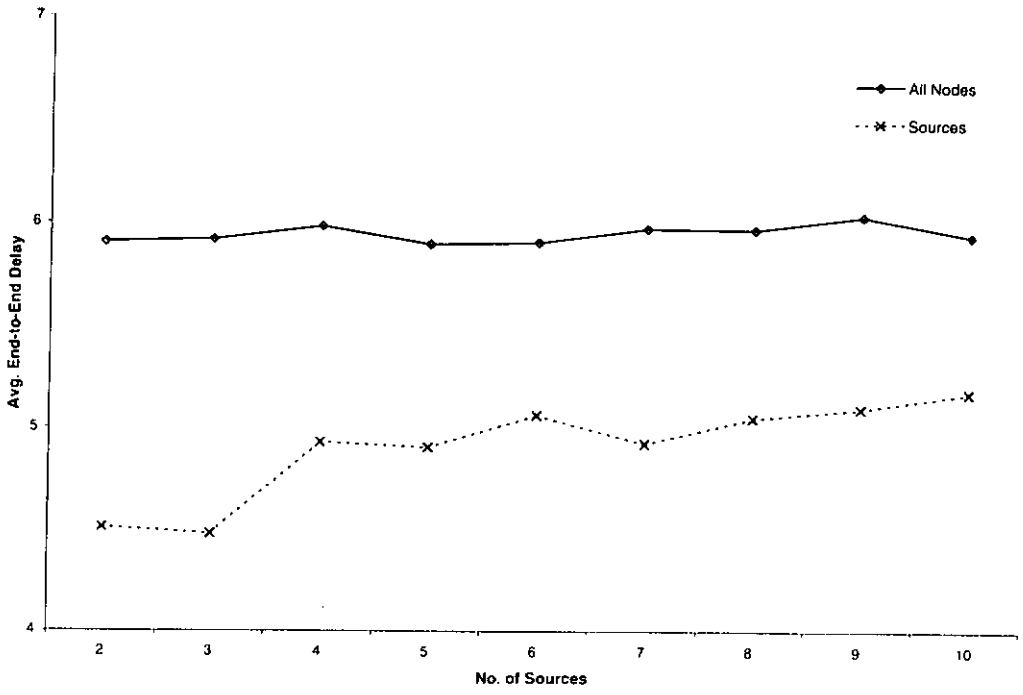
(a) Scenario 1



(b) Scenario 2



(c) Scenario 3



(d) Scenario 4

Fig. 3.1. End-to-end performance in different location of the core router.

Minimal Control Overhead

In the source-specific approaches such as DVMRP and PIM-DM, a large amount of control messages are introduced over the construction of a multicast distribution tree because all routers in the network have to periodically generate and process control messages for every multicast group regardless of whether or not they belong to the multicast tree of the group. Unlike these approaches, ASCORT employs an approach similar to what used in PIM-SM so as to reduce the number of control messages. In PIM-SM, each member router sends a *Join/Prune* message toward the RP. Routers along the shortest path from the member router to the RP receive the message. Similarly, ASCORT requires each member router to send a graft message toward its selected source and each first-hop router sends a graft message toward the core. Routers lying on the shortest path from the member router to its selected source and routers lying on the shortest path from the first-hop router to the core receive the message. The readers should note that only the routers on the multicast tree for the group are required to generate and process control messages. Those routers, which do not belong to the multicast tree, are not involved in the generation and processing of control messages.

Adaptability of Network Dynamics

To deal with the failures of routers and network links, ASCORT employs soft state mechanism to adapt to network dynamics. Similar to PIM-SM which also makes use of soft-state mechanism, ASCORT associates each incoming and outgoing interface with a timer. The timer associated with an interface will be reset on the receipt of a graft message or a multicast packet via the interface. When the timer associated with an interface is expired, the interface is removed from the multicast forwarding entry.

Adaptability of Group Dynamics

Unlike PIM-DM in which periodic broadcast and prune is used, ASCORT employs explicit graft and prune to handle group dynamics. In PIM-DM, a timer is associated with each interface in a router. The timer of an interface resets when a *Prune* message is received from that interface. When the timer of an interface expires, multicast packets are forwarded to that interface. As a result, when a host joins a multicast group, it may wait for some time until the timers expire before it can receive multicast packets from the sources.

In order to reduce the latency for member join and leave, ASCORT employs explicit graft and prune. When a host joins the multicast group, it sends a graft message toward the base source so that it is attached to the reduced tree rooted at the base source. On the other hand, when a member leaves the multicast group, it sends a prune message toward the selected source so as to remove the tree branch connecting the member on the reduced tree.

Loop Freedom

In ASCORT, the incoming interface in a multicast forwarding entry is determined by using RPF. The use of RPF to identify the incoming interface prohibits the formation of loop when no loop is formed in the underlying unicast routing. As a result, the multicast distribution trees constructed by ASCORT is free of any loop.

Independence of Unicast Routing Protocols

In ASCORT, multicast distribution trees are built by RPF lookup in the unicast routing tables. However, ASCORT does not depend on any specific unicast routing protocol in the computation of routes.

Support for Quality of Service

Multicast distribution trees constructed by ASCORT do not establish any tunnel. This allows ASCORT to support QoS.

3.3 ASCORT in Details

3.3.1 Messages

ASCORT uses five messages to construct and maintain a multicast distribution tree. They are *GRAFT*( ), *PRUNE*( ), *MODIFY*( ), *UPDATE*( ) and *CLEAR*( ). Their functions are summarized in Table 3.1.

Messages	Sent by	Removed by	Function
GRAFT(s_i, g) or *GRAFT*(*, g)	Member Routers First-Hop Routers	The first-hop router of s_i The core.	Build a tree branch connecting a member router to a reduced tree. Build a tree branch connecting a first-hop router to a bi-directional tree.
PRUNE(s_i, g)	Leaf Routers	Routers with more than one directly connected downstream routers or Member Routers.	Prune away a tree branch on the reduced tree rooted at s_i.
MODIFY(s_i, s_k, g)	Member Routers	Any on-tree Routers.	Change the source address from s_i to s_k in a source-specific multicast forwarding entry.
UPDATE(s_i, g)	Core	First-Hop Routers.	Update base source information.
CLEAR(g)	First-Hop Routers	Any routers.	Remove multicast forwarding entries and base source information for multicast group g.

Table 3.1. Messages for ASCORT.

3.3.2 Multicast Forwarding Entry

The multicast distribution tree constructed by ASCORT is composed of a forest of reduced trees and a bi-directional tree. Each router is required to maintain only one multicast forwarding entry for a given multicast group. In ASCORT, there are two types of multicast forwarding entries: *source-specific multicast forwarding entries* and *wildcard multicast forwarding entries*. Source-specific multicast forwarding entries are maintained at routers on reduced trees whereas wildcard multicast forwarding entries are maintained in routers on the bi-directional tree.

A source-specific multicast forwarding entry is in the form of (s_i, g, in_i, out_i) for the source s_i and the multicast group g. Specifically, the incoming interface in_i is the upstream router from which the router receives multicast packets and the outgoing interface list out_i contains the interfaces to which the packets for the multicast group g are forwarded.

On the other hand, a wildcard multicast forwarding entry is in the form of (*, g, *in*, *out*) where * is a *wildcard*, g is the multicast group, *in* is the incoming interface list, and *out* is the outgoing interface list. The incoming interface list *in* contains the interfaces from which packets for the multicast group g are received whereas the outgoing interface list *out* contains the interfaces to which packets for the multicast group g are forwarded. Unlike source-specific multicast forwarding entries, a wildcard multicast forwarding entry may contain several incoming interfaces instead of only one incoming interface.

Similar to PIM-SM, soft-state mechanism is employed to adapt to network changes in ASCORT. Each interface including incoming and outgoing maintained in the multicast forwarding entry including source-specific and wildcard is associated with a timer. The value of the timer can be set to a fixed value just as what PIM-SM does. As an alternative, the value of the timer can be set to a variable value in order to adapt to the volume of control traffics and available bandwidth on network links [Sharma *et al.* 1997]. The usage of timers will be presented in Section 3.3.9.

3.3.3 Multicast Forwarding Algorithm

The forwarding algorithm for source-specific multicast forwarding entries and that for wildcard multicast forwarding entries are different in ASCORT. When a router which maintains a source-specific multicast forwarding entry, (s_i, g, in_i, out_i), receives a multicast packet to the multicast group g via interface $\{v\}$, it checks whether $\{v\} = in_i$. If so, the router forwards the packet to all interfaces in out_i. Otherwise, the router discards the packet. In case that a router which maintains a wildcard multicast forwarding entry, (*, g, *in*, *out*), receives a multicast packet to the multicast group g via interface $\{v\}$, it checks whether $\{v\} \in in$. If so, the router forwards the packet to all interfaces in *out* except $\{v\}$. Otherwise, the router discards the packet.

The forwarding algorithm for source-specific multicast forwarding entries in ASCORT is different from that for existing source-specific tree approach such as PIM-DM. In PIM-DM, when a router which maintains a source-specific multicast forwarding entry receives a multicast packet, it forwards the packet to all interfaces in the outgoing interface list if (i) the entry's source address prefix is the longest initial bit-substring (among all other multicast forwarding entries) that matches the packet's source address; (ii) the entry's group address prefix is the longest initial bit-substring that matches the packet's group address; and (iii) the interface from which the packet was received is equal to its incoming interface. Unlike PIM-DM, ASCORT does not require routers to perform the longest match on the source address prefix. In ASCORT, when a router which maintains a source-specific multicast forwarding entry receives a multicast packet, it forwards the packet to all interfaces in the outgoing interface list if the group address prefix in the entry is the longest initial bit-substring that matches the packet's group address, and the interface from which the packet was received is equal to the incoming interface in the entry.

The forwarding algorithm for wildcard multicast forwarding entry in ASCORT is different from that for existing shared tree approaches such as PIM-SM. Specifically, wildcard multicast forwarding entries in PIM-SM are uni-directional whereas those in ASCORT are bi-directional. In order to implement the bi-directional function in a wildcard multicast forwarding entry in ASCORT, a set of incoming interfaces is contained in a wildcard multicast forwarding entry instead of only one interface is contained in a uni-directional wildcard multicast forwarding entry in PIM-SM. By allowing a wildcard multicast forwarding entry to contain a set of incoming interfaces, bi-directional functionality can be implemented using its forwarding algorithm in ASCORT.

To show how the bi-directional functionality can be implemented, let us consider that a router which maintains a uni-directional multicast forwarding entry (*, g, $\{a\}$, $\{b\}$). In this entry, there is only one incoming interface $\{a\}$ and one outgoing interface $\{b\}$. To make this entry becomes bi-directional (i.e. allow a packet receive from $\{a\}$ and then forward to $\{b\}$ and also allow a packet receive from $\{b\}$ and then forward to $\{a\}$), the entry should contain $\{a, b\}$ as incoming interfaces and $\{a, b\}$ as outgoing interfaces. By applying the forwarding algorithm to a wildcard multicast forwarding entry (*, g, in, out) where $in = \{a, b\}$ and $out = \{a, b\}$, a packet can be traversed in both directions. When the router receives a multicast packet for multicast group g via interface $\{a\}$, it checks whether $\{a\} \in in$. If so, it forwards the packet to $out - \{a\} = \{b\}$. Similarly, when the router receives a multicast packet for multicast group g via interface $\{b\}$, it checks whether $\{b\} \in in$. If so, it forwards the packet to $out - \{b\} = \{a\}$.

In this example, although the incoming interface list and the outgoing interface list are the same, both of them are required in the entry because the incoming interface list may be different from the outgoing interface list. For example, it is possible that a first-hop router maintains a

wildcard multicast forwarding entry such that the interface connecting the sender is included in the incoming interface but is not included in the outgoing interface list. Another example is that a router connects to a set of downstream routers on the reduced tree, R_1, and another set of routers on the bi-directional tree, R_2. Let us note that downstream routers on the reduced trees are intended to receive multicast packets only. It is for this reason that we are required to add the interface connecting R_1 into the outgoing interface list only. On the other hand, the wildcard multicast forwarding entry maintained at the router includes the interfaces connecting routers in $R_1 \cup R_2$ in the outgoing interface list but only includes the interfaces connecting routers in R_2 in the incoming interface list.

3.3.4 Construction of Multicast Distribution Tree

3.3.4.1 The First Host Sending to the Multicast Group

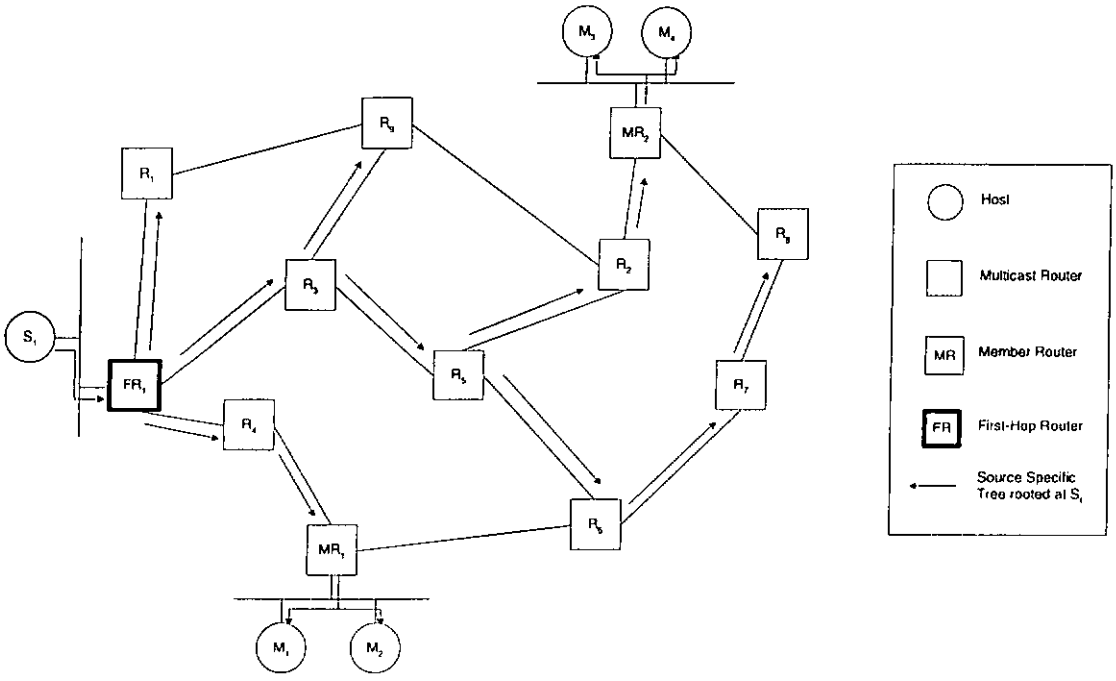
Let s_b be the first host which sends data packets to the multicast group g. In ASCORT, s_b is called *base source* and its first-hop router, fr_b , is called *core*. The base source s_b broadcasts its first data packet to the multicast group g. Upon receiving the packet, a router creates a source-specific multicast forwarding entry for s_b and g. In order to create such source-specific multicast forwarding entry for s_b and g, ASCORT makes use of Reverse Path Forwarding (RPF) lookup [Deering 1991] in the unicast routing table to identify the incoming interface which is on the shortest path to the source. On the other hand, the outgoing interface list out_b in the source-specific multicast forwarding entry contains interfaces that have multicast routers present or host members for the multicast group g. This results in the construction of a source-specific multicast tree rooted at s_b connecting all multicast routers in the network. To construct a source-specific tree connecting all group members, each member router (i.e. a designated router of a LAN in which there is at least a member of the multicast group g) sends a $GRAFT(s_b, g)$ message toward s_b. Upon receiving a $GRAFT(s_b, g)$ message from interface $\{v\}$, a router, which maintains a source-specific multicast forwarding entry (s_b, g, in_b, out_b), forwards the message towards s_b and resets its timer for $\{v\}$. In ASCORT, a timer for an outgoing interface maintained in a source-specific multicast forwarding entry resets when a $GRAFT(s_b, g)$ message is received from that interface before the timer timeout. On the other hand, when the timer for each outgoing interface maintained in a source-specific multicast forwarding entry expires, the entry will be deleted. As a result, only routers on the branches connecting members to s_b receives $GRAFT(s_b, g)$ messages and hence their entries are updated. On the contrary, the branches which have no member connected will be removed since the routers on the branches will not receive any $GRAFT(s_b, g)$ message and hence their entries will be expired and then deleted. Consequently, a source-specific tree rooted at s_b connecting all members is constructed. Unlike DVMRP and PIM-DM which require leaf

routers to send prune messages upstream to remove tree branches, ASCORT requires member routers to send *GRAFT*(s_b, g) messages upstream to build tree branches.

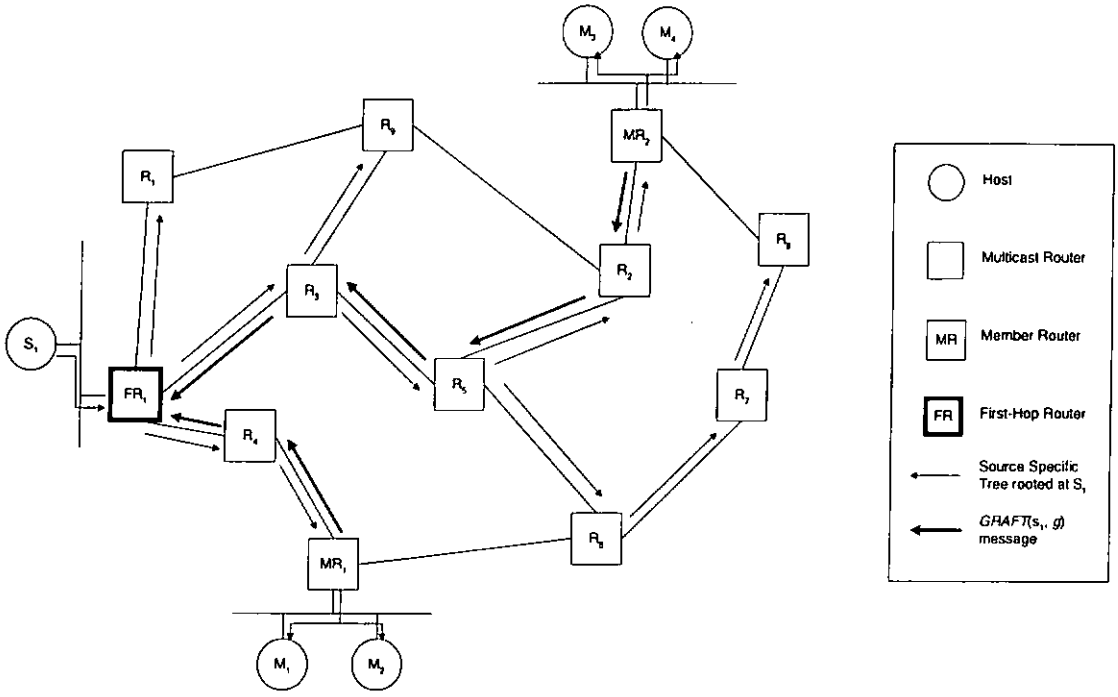
Furthermore, in ASCORT, the broadcasting of the first data packet makes each multicast router maintain the base source information which is used for a new source to build bi-directional route and for a new member to join the reduced tree rooted at the base source. Unlike PIM-SM in which *Bootstrap Router* (BSR) broadcasts a bootstrap message to provide RP information to all routers in the network, ASCORT makes use of data packet to provide base source information to all routers in the network. As a result, no control message is required to provide base source information in ASCORT.

Similar to PIM-SM in which each member router performs periodic refresh by sending a *Join/Prune* message toward a RP periodically, ASCORT performs periodic refresh by requiring each member router to send a *GRAFT*(s_b, g) message periodically toward the base source to maintain a multicast distribution tree.

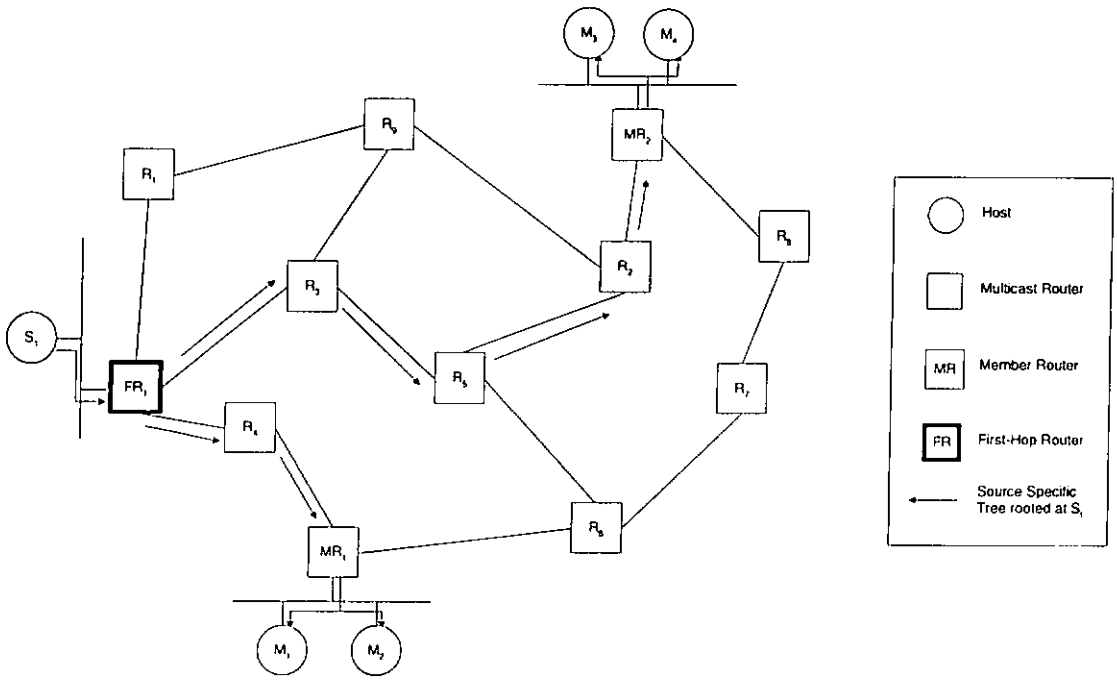
To illustrate how ASCORT constructs a source-specific tree connecting all members, an example is given in Fig. 3.2. In the example, there is a single source s_1 in a multicast group g and there are four members (i.e. M_1, M_2, M_3 and M_4). We assume a source-specific tree rooted at s_1 connecting all multicast routers in the network is constructed after s_1 broadcasts its first multicast packet and each multicast router performs RPF to identify the incoming interface, which is on the shortest path to s_1, to receive s_1's packets.



(a) The source-specific tree rooted at s_1 connecting all multicast routers.



(b) The operations of ASCORT.



(c) A source-specific tree rooted at s_1 connecting group members.

Fig. 3.2. An example to construct source-specific trees by ASCORT.

Fig. 3.2(a) shows a source-specific tree rooted at s_1 connecting all multicast routers in the network. To construct a source-specific tree connecting all group members, both MR_1 and MR_2 sends a $GRAFT(s_1, g)$ message toward s_1 (Fig. 3.2(b)). Upon receiving a $GRAFT(s_1, g)$ message, a router updates its source-specific multicast forwarding entry. As a result, the source-specific multicast forwarding entries maintained at the routers along the path from MR_2 to FR_1 are updated. Specifically, the source specific multicast forwarding entries maintained in MR_2, R_2, R_5, R_3, and FR_1 are updated. Similarly, the source-specific multicast forwarding entries maintained in MR_1, R_4, and FR_1 are updated. Nevertheless, the source-specific multicast forwarding entries maintained at R_1, R_6, R_7, R_8, and R_9 are expired and then deleted since these routers did not generate or receive any $GRAFT(s_1, g)$ message. Consequently, a source-specific tree rooted at s_1 connecting all group members is constructed as shown in Fig. 3.2(c).

3.3.4.2 Additional Hosts Sending to the Multicast Group

In the previous section, we have considered the case that there is a single source in a multicast group. Realistically, more than one source may join a multicast group. Similar to the case for the first host sending to a multicast group g, when a new source s_i where $s_i \neq s_b$ joins the same multicast group, s_i broadcasts its first multicast packet and member routers send $GRAFT(s_i, g)$ messages toward s_i to construct a source-specific tree. Furthermore, the first hop router of s_i, fr_i, sends a $GRAFT(*, g)$ message toward the core and each member router selects a source to join. This triggers the construction of bi-directional tree and reduced trees. It is important to note that multicast routers in the network are not required to keep the information about s_i. Nevertheless, each multicast router is required to store the base source s_b information.

In the following sections, we provide how ASCORT constructs the bi-directional tree rooted at s_b and the reduced tree rooted at s_i.

3.3.4.2.1 Construction of the Bi-Directional Tree

A bi-directional tree is a tree rooted at the base source s_b which connects all the first-hop routers together by building a bi-directional route between the core and the first-hop routers. A bi-directional route is a route which allows multicast traffics to traverse from one end to the other and vice versa. Each router on the bi-directional tree maintains a wildcard multicast forwarding entry.

To construct a bi-directional tree, the first-hop router fr_i of a new source s_i sends a $GRAFT(*, g)$ message toward the base source s_b via interface $\{v\}$ and performs the following procedures.

1. If fr_i already maintains a wildcard multicast forwarding entry, $(*, g, in, out)$, it updates $in = in \cup \{$the interface connecting the router and $s_i\}$ and $out = out \cup \{v\}$.

2. Else if fr_i already maintains a source-specific multicast forwarding entry, (s_i, g, in_i, out_i), it replaces the entry with a new wildcard multicast forwarding entry, (*, g, in, out) where $in = in_i \cup \{v\}$ and $out = out_i \cup \{v\}$.

After performing the above procedures, the first-hop router fr_i maintains a wildcard multicast forwarding entry. Furthermore, all routers lying on the path from the first-hop router fr_i to the base source s_b are required to maintain a wildcard multicast forwarding entry. To establish such entry, a router performs the following grafting procedures upon receiving a *GRAFT*(*, g) message via interface $\{v\}$.

1. If the router is the core, it performs the following procedures.

   1.1. If the router maintains a wildcard multicast forwarding entry, (*, g, in, out), it will set $in = in \cup \{v\}$ and $out = out \cup \{v\}$.

   1.2. Else if the router maintains a source-specific multicast forwarding entry (s_b, g, in_b, out_b), it will replace the source-specific multicast forwarding entry by a new wildcard multicast forwarding entry, (*, g, in, out) where $in = in_b \cup \{v\}$ and $out = out_b \cup \{v\}$.

2. Else if the router is not the core, it performs the following procedures.

   2.1. If the router does not maintain any source-specific or wildcard multicast forwarding entry for the multicast group g, it forwards the *GRAFT*(*, g) message toward the base source s_b. The router then creates a new wildcard multicast forwarding entry, (*, g, in, out) where $in = out =$ {the interface to which the *GRAFT*( ) message is sent} $\cup \{v\}$.

   2.2. Else if the router already maintains a wildcard multicast forwarding entry, (*, g, in, out), it forwards the *GRAFT*(*, g) message toward the base source and set $in = in \cup \{v\}$ and $out = out \cup$ {the interface to which the *GRAFT*( ) message is sent} $\cup \{v\}$.

   2.3. Else if the router already maintains a source-specific multicast forwarding entry, (s_i, g, in_i, out_i), it creates a new wildcard multicast forwarding entry, (*, g, in, out) where $in =$ {the interface to which the *GRAFT*( ) message is sent} $\cup \{v\}$ and $out = out_i \cup$ {the interface to which the *GRAFT* message is sent} $\cup \{v\}$. The router

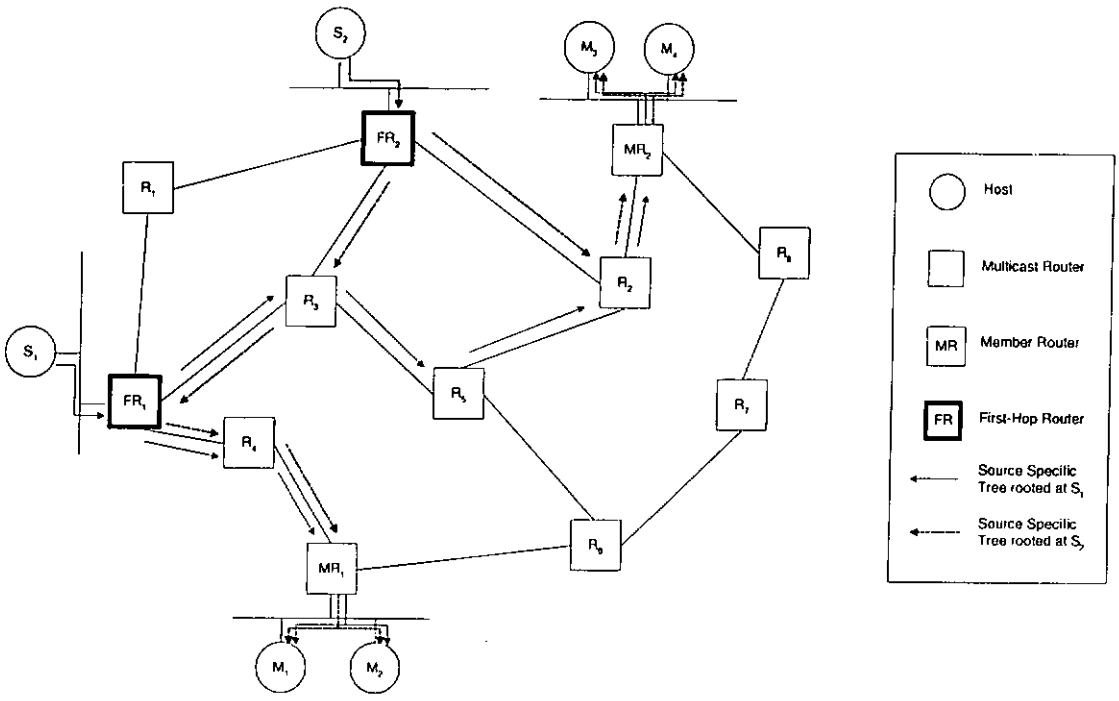
then forwards the *GRAFT*(*, g) message toward the base source s_b and deletes the existing source-specific multicast forwarding entry.

Furthermore, in order to eliminate any loop in the bi-directional tree, the router sends a *PRUNE*(s_i, g) message to in_i if $in_i \neq \{v\}$ and $in_i \neq$ {the interface to which the *GRAFT*( ) message is sent}. The way that a router responds on the receipt of a *PRUNE*( ) message will be given in Section 3.3.7.3.

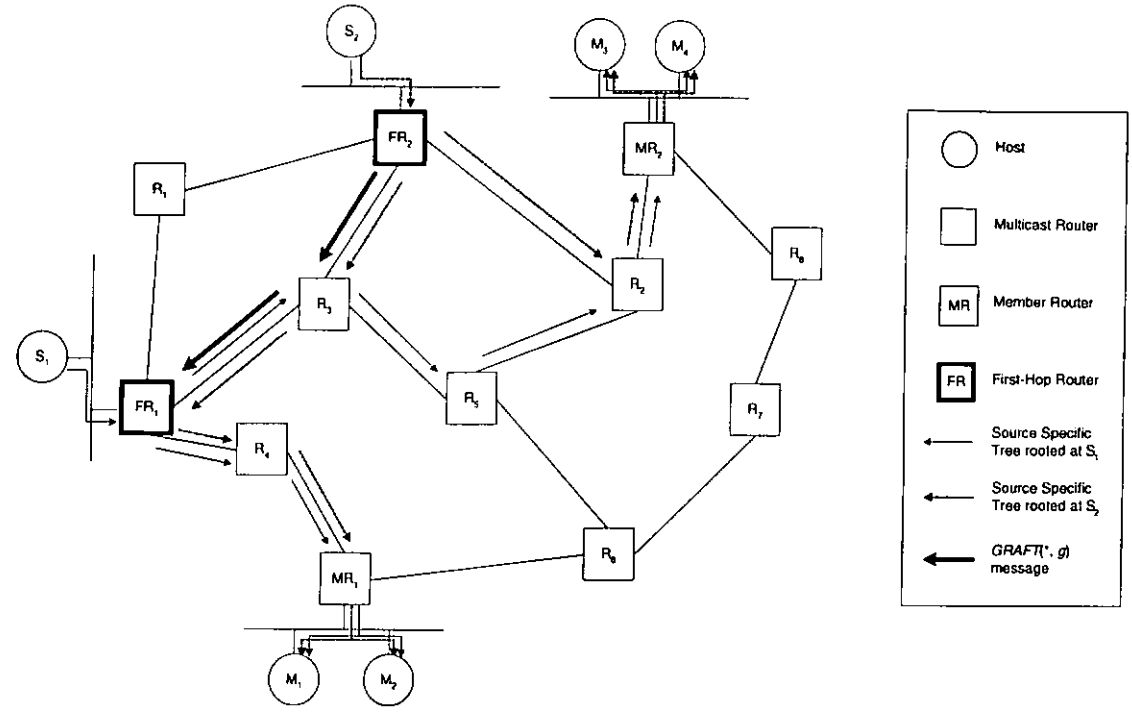
In addition to create/ modify the multicast forwarding entry, the router resets the timer associated with $\{v\}$ and the interface to which the *GRAFT*(*, g) message is sent.

When all the routers along the path from the first-hop router fr_i to the base source s_b have performed the above procedures, a bi-directional route between fr_i and s_b is established. In other words, fr_i is attached to the bi-directional tree rooted at s_b.

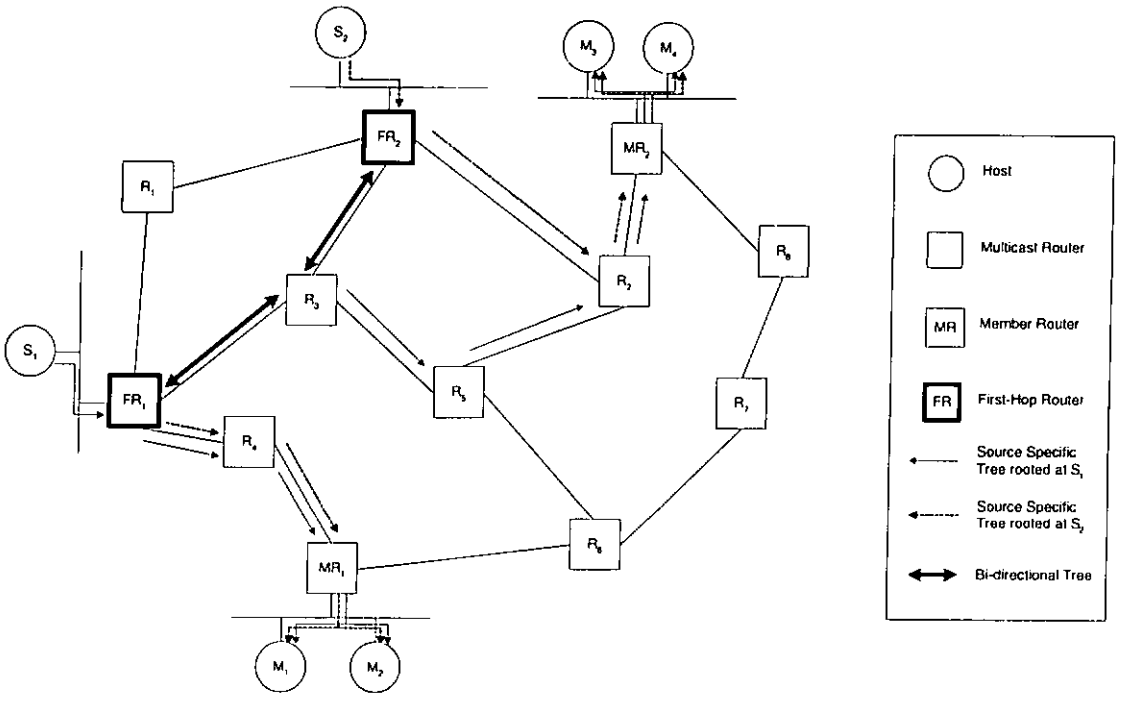
An example is given in Fig. 3.3 to illustrate how ASCORT construct a bi-directional route. In the example, there are two sources, s_1 and s_2, and four members, M_1, M_2, M_3, and M_4. We assume two source specific trees are already constructed.



(a) The source-specific trees rooted at s_1 and s_2 respectively.



(b) The operations of ASCORT.



(c) Bi-directional tree constructed by ASCORT.

Fig. 3.3. An example to construct a bi-directional tree by ASCORT.

Two source-specific trees are constructed connecting the same set of members (Fig. 3.3(a)). One is rooted at s_1 and the other one is rooted at s_2. ASCORT requires FR_2 to send a *GRAFT*(*, *g*) message toward FR_1 so as to build a bi-directional route as shown in Fig. 3.3(b). This results in a wildcard multicast forwarding entry is created in FR_2, R_3, and FR_1 after sending or receiving a *GRAFT*(*, *g*) message. As a result, a bi-directional route is built connecting FR_1 and FR_2 (Fig. 3.3(c)). Consequently, multicast packets send from s_1 can traverse FR_1, R_3, FR_2, R_2, and MR_2 to reach M_3 and M_4. Similarly, multicast packets send from s_2 can traverse FR_2, R_3, FR_1, R_4, and MR_1 to reach M_1 and M_2.

3.3.4.2.2 Construction of Reduced Trees

For instance, there are two source-specific trees rooted at s_b and s_i connecting the same set of members. Each member router mr_j, maintains two source-specific multicast forwarding entries, (s_b, *g*, in_b, out_b) and (s_i, *g*, in_i, out_i). Recall that a reduced tree is a tree rooted at a source connecting a subset of members, the source-specific tree rooted at s_b connecting all members is an extreme case for a reduced tree. To generalize the case, we assume a forest of reduced trees for a multicast group *g* constructed by ASCORT already exists. Each member router mr_j should be already on one of the reduced trees rooted at s_k and maintains a source specific multicast forwarding entry (s_k, *g*, in_k, out_k). When a new source s_i joins the multicast group *g*, a source-specific tree rooted at s_i connecting all members is constructed. As a result, each member router mr_j, now maintains two source-specific multicast forwarding entries, (s_k, *g*, in_k, out_k) and (s_i, *g*, in_i, out_i) where $s_k \neq s_i$. To construct a reduced tree, mr_j carries out the following operations.

1. mr_j selects the selected source from s_k and s_i. The selection is based on the hop counts from mr_j to s_k and s_i. The one with the smallest number of hop counts will be chosen as the selected source. In case that the hop counts from mr_j to s_k and s_i are the same, the one with the smallest IP address will be chosen.

   In ASCORT, each source has to set the multicast packet's Time to Live (TTL) to the same value, say 32 or 64. When a member router receives a multicast packet from a source s_i, it can calculate the hop count by subtracting the received packet's TTL value from the original value of TTL of the packet sent by s_i.

2. If mr_j can receive s_i's multicast packet from in_k or it can receive s_k's multicast packet from in_i, it will send a *GRAFT*( ) message toward the selected source only when mr_j performs periodic refresh. It is important to note that mr_j is required to send a *GRAFT*( ) message toward both s_i and s_k when mr_j performs periodic refresh before one of the above conditions is satisfied. The transmission of a *GRAFT*( ) message to the selected source resets the timers for a multicast forwarding entry maintained in the routers along the tree branch connecting mr_j to the selected

source and hence re-builds the tree-branch. On the contrary, the lack of sending a *GRAFT( )* message to the source, which has not been chosen as the selected source, causes the source-specific multicast forwarding entries maintained in the routers lying on the branch connecting mr_j to the source expires and they will be removed. As a result, the branch connecting mr_j to the source, which has not been chosen, is destroyed. The way that a router responds on the receipt of a *GRAFT*(s_i, g) message will be given in the Section 3.3.6.1.

The two conditions (i.e. mr_j can receive s_i's multicast packet from in_k and mr_j can receive s_k's multicast packet from in_i) mentioned in the above step is used to indicate whether the bi-directional route between s_i and the base source s_b is established successfully. The satisfaction of one of the conditions guarantees the successful establishment of the bi-directional route. ASCORT requires establishment of bi-directional route between s_i and s_b before destroying the branch connecting mr_j to the source, which has not been chosen as the selected source, because we want to ensure no packet loss will be introduced by ASCORT once a member has joined a multicast group. This makes ASCORT suitable for multicast applications which require reliable packet delivery and cannot tolerate packet loss.

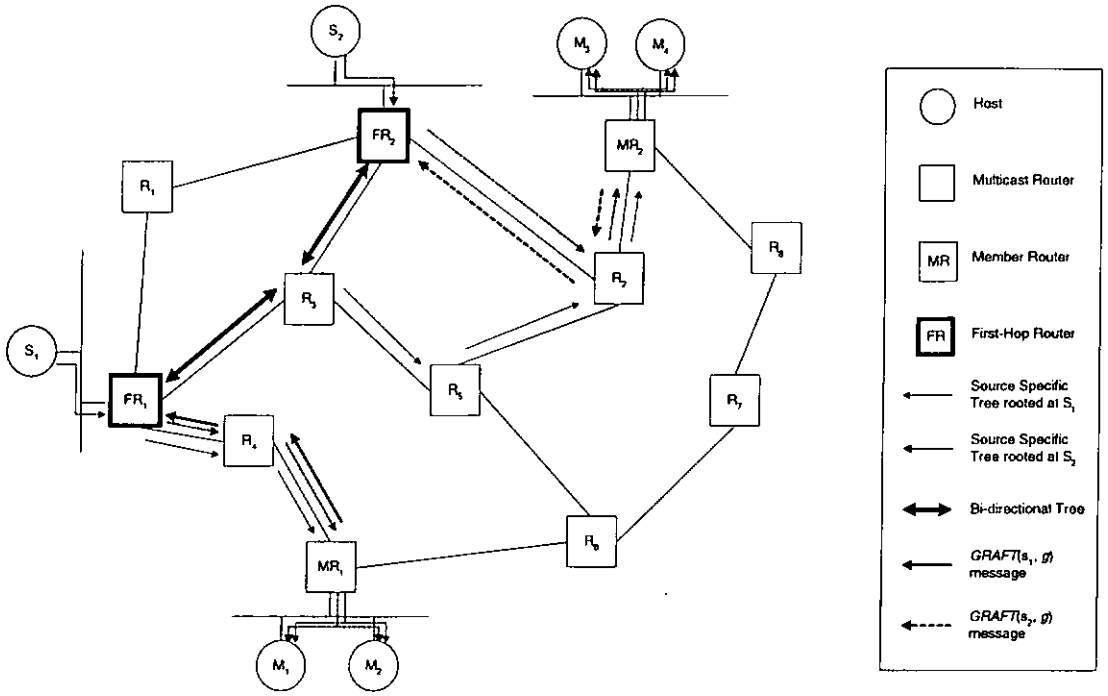
It is important to note that there is an alternative for multimedia applications which can tolerate certain degree of packet loss. For these applications, each member router mr_j destroys the tree branches connecting mr_j to the source, which has not been chosen as the selected source, once it has chosen its selected source. However, we cannot guarantee the bi-directional route between s_i and the base source s_b is established before the reduced tree is constructed. In this case, some members may suffer from packet loss for a short period of time.

The fact that each member router performs the above procedures results in partitioning group members into different disjoint subsets. Consequently, a forest of reduced trees, each of which is rooted at a source and connects to a subset of group members, is constructed. Let us note that there is no overlapping between reduced trees constructed by ASCORT. The proof for no overlapping between reduced trees will be given in Section 3.4.

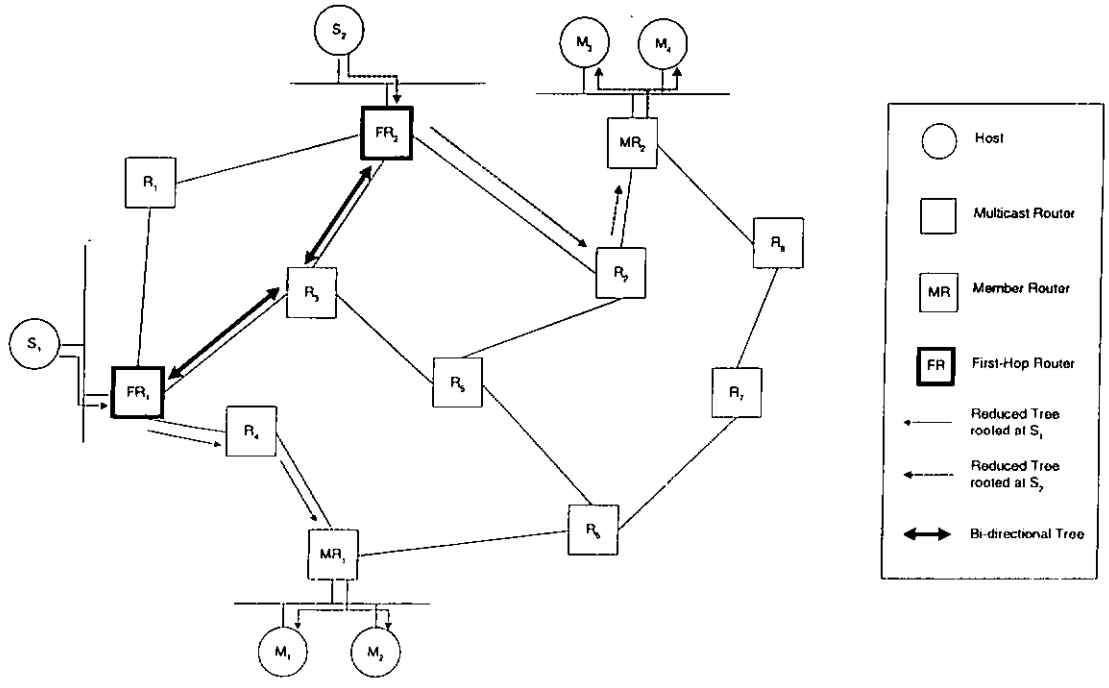
Let us further note that only one multicast forwarding entry is kept in each router on the multicast distribution tree constructed by ASCORT. Specifically, a source specific multicast forwarding entry is maintained in the routers on reduced trees and a wildcard multicast forwarding entry is maintained in the routers on the bi-directional tree. In case that the bi-directional tree overlapped with a reduced tree (e.g. a first-hop router is on a reduced tree and a bi-directional tree at the same time), only a wildcard multicast forwarding entry is maintained in the routers along the overlapping branches.

To illustrate how reduced trees are constructed, an example is provided in Fig. 3.4. In this example, there are two sources and four members. We assume the bi-directional route connecting

FR_1 and FR_2 is already established. We further assume MR_1 selects s_1 as its selected source and MR_2 selects s_2 as its selected source.



(a) The operations of ASCORT.



(b) Reduced trees constructed by ASCORT.

Fig. 3.4. An example to construct reduced trees by ASCORT.

Since MR_1 selects s_1 as its selected source and MR_2 selects s_2 as its selected source, MR_1 sends a $GRAFT(s_1, g)$ message to FR_1 and MR_2 sends a $GRAFT(s_2, g)$ message toward FR_2. As a result, the source-specific multicast forwarding entries for s_1 maintained at MR_1 and R_4 and the source-specific multicast forwarding entries for s_2 maintained at MR_2 and R_2 are updated. In contrast, the source specific multicast forwarding entries for s_1 maintained at MR_2 and R_2 are expired and then deleted. Similarly, the source specific multicast forwarding entries for s_2 maintained in MR_1 and R_4 are expired and then deleted. As a consequence, two reduced trees are constructed: one is rooted at s_1 and the other one is rooted at s_2.

3.3.4.3 Some Variations

In the previous sections, the construction of reduced trees and the bi-directional tree is triggered by a new source sending to a multicast group. However, ASCORT is flexible enough to be applied on top of a set of source-specific trees which already exist. The source-specific trees are not required to be constructed by ASCORT, they can be constructed by any source-specific approach such as DVMRP and PIM-DM. To construct reduced trees and the bi-directional tree from a set of source-specific trees, each multicast router in the network is required to maintain a threshold for a multicast group g. When the number of source-specific multicast forwarding entries for the multicast group g maintained in a member router is larger than the threshold, member routers start to construct reduced trees. Similarly, first-hop routers construct bi-directional tree when the number of source-specific multicast forwarding entries for the multicast group g is larger than the threshold. Let us note that the base source in this case will be the source with the smallest IP address in the multicast group instead of the first source which sends to the multicast group. To construct reduced trees and the bi-directional tree, member routers and first-hop routers should perform the procedures mentioned in Section 3.3.4.2.1 and Section 3.3.4.2.2.

3.3.5 Packet Dissemination in the Multicast Distribution Tree

When all routers on the distribution tree constructed by ASCORT have maintained a multicast forwarding entry, a packet can be delivered from all sources to all members. To illustrate how packet dissemination is performed in ASCORT, we make use of Fig. 3.4(b) as an example.

In this example, there are two reduced trees. The first reduced tree rooted at s_1 connecting MR_1 and the second reduced tree rooted at s_2 connecting MR_2. In addition to the reduced trees, a bi-directional tree is constructed to connect FR_1 and FR_2 together. In Fig. 3.4(b), each of MR_1, MR_2, R_2, and R_4 maintains a source-specific multicast forwarding entry whereas each of FR_1, FR_2, and R_3 maintains a wildcard multicast forwarding entry.

When s_1 disseminates a multicast packet, its first-hop router, FR_1, will send a copy to FR_2 via the bi-directional route in addition to its reduced tree. Upon receiving the packet, FR_2 will

forward this packet to its reduced tree. Similarly, when s_2 disseminates a multicast packet, its first-hop router, FR_2, will send a copy to FR_1 via the bi-directional route in addition to its reduced tree. Upon receiving the packet, FR_1 will forward this packet to its reduced tree. This allows all the members to receive the multicast packet (M_1 and M_2 receive a copy on the reduced tree rooted at s_1; and M_3 and M_4 receive a copy on the reduced tree rooted at s_2). This shows how a multicast packet disseminated by a source can reach all the members in the multicast group.

3.3.6 Static Maintenance of Multicast Distribution Trees

Due to the dynamic nature of multicast groups, it is possible that (i) an existing source is still active; (ii) an existing source becomes inactive; (iii) a new source joins a multicast group; (iv) an existing member still wants to join the multicast group; (v) an existing member leaves its multicast group; and (vi) a new member joins a multicast group. We refer case (i) and case (iv) as static cases to maintain a multicast distribution tree whereas case (ii)-(iii) and case (iv)-(v) as dynamic cases to maintain a multicast distribution tree.

In this Section, we provide the details how ASCORT deals with case (i) and case (iv). After that, we will discuss how ASCORT deals with case (i)-(ii) and case (iv)-(v) in next section.

3.3.6.1 Maintenance of Reduced Trees

In ASCORT, each member router mr_j sends a $GRAFT(s_i, g)$ message toward its selected source s_i periodically if it still wants to receive packets in the multicast group g and the timer associated with the incoming interface does not expire yet. Upon receiving the $GRAFT(s_i, g)$ message via interface $\{v\}$, a router performs the following operations.

1. If the router does not maintain any source-specific or wildcard multicast forwarding entry for the multicast group g, it creates a new source-specific multicast forwarding entry, (s_i, g, in_i, out_i) where in_i is the interface to which the $GRAFT(\ )$ message is forwarded and $out_i = \{v\}$. The router then forwards the $GRAFT(s_i, g)$ message toward s_i.

2. Else if the router maintains a wildcard multicast forwarding entry, $(*, g, in, out)$, it sends a $GRAFT(*, g)$ message toward s_i. Furthermore, the router sets $in = in \cup \{$the interface to which the message is sent$\}$ and $out = out \cup \{v\}$. The way how a router responds on the receipt of a $GRAFT(*, g)$ message is given in Section 3.3.4.2.1.

3. Else if the router maintains a source-specific multicast forwarding entry, (s_i, g, in_i, out_i), it forwards the $GRAFT(s_i, g)$ message toward s_i. Furthermore, the router modifies the outgoing interface list $out = out \cup \{v\}$ if $\{v\} \notin out$. The router also changes in to the interface to which the $GRAFT(\ )$ message is forwarded if in is different from that interface.

4. Else if the router maintains a source-specific multicast forwarding entry, (s_k, g, *in*, *out*) where $s_i \neq s_k$, it sends a *GRAFT*(s_k, g) message toward s_k. Furthermore, it sets *out* = *out* $\cup$ $\{v\}$. The router also changes *in* to the interface to which the *GRAFT*( ) message is sent if *in* is different from that interface. After updating the forwarding entry, the router sends a *MODIFY*(s_i, s_k, g) message to the interface $\{v\}$. Upon receiving a *MODIFY*(s_i, s_k, g) message, a router performs the following procedures.

   4.1. If the router already maintains a source-specific multicast forwarding entry, (s_i, g, *in*, *out*), it changes the entry from (s_i, g, *in*, *out*) to (s_k, g, *in*, *out*) and then forwards the *MODIFY*(s_i, s_k, g) message to all interfaces in *out*.

   4.2. Otherwise, the router simply discards the *MODIFY*( ) message.

In addition to create/ modify the multicast forwarding entry, the router resets the timer associated with $\{v\}$.

In case that s_i becomes inactive, the timer associated with the incoming interface expires. If mr_j still wants to receive packets in the multicast group g, mr_j sends a *GRAFT*(s_b, g) message toward the base source s_b instead of sending a *GRAFT*(s_i, g) toward s_i. This connects mr_j to the reduced tree rooted at the base source s_b. Since the routers along the path from mr_j to s_i cannot receive the *GRAFT*(s_i, g) message before timeout, their multicast forwarding entries will be deleted. This results in the destruction of the branch of the reduced tree rooted at s_i connecting mr_j. As a result, the member m_j receives multicast packets on the reduced tree rooted at the base source s_b thereafter.

3.3.6.2 Maintenance of Bi-Directional Tree

The first-hop router of each source sends a *GRAFT*(*, g) message toward the base source s_b periodically if the source is still active in the multicast group g. On the receipt of the *GRAFT*(*, g) message via interface $\{v\}$, a router performs the grafting procedures mentioned in Section 3.3.4.2.1.

3.3.6.3 Maintenance of Base Source Information

The core, fr_b, multicasts an *UPDATE*(s_b, g) message periodically to the first-hop routers of active sources. The core fr_b considers a source active if it receives a multicast packet from that source before timeout. If a first-hop router cannot receive an *UPDATE*(s_b, g) message before timeout, this implies the base source has left and/ or the core is failed. In this case, the first-hop router deletes

the multicast forwarding entry and the base source information for the multicast group g. Furthermore, the first-hop router sends a *CLEAR*(g) message to all its interfaces.

Upon receiving a *CLEAR*(g) message via interface $\{v\}$, if a router maintains any multicast forwarding entry and/or base source information for the multicast group g, it deletes the multicast forwarding entry and/or the base source information. The router then forwards the *CLEAR*(g) message to all its interfaces except $\{v\}$. Otherwise, if the router does not maintain any multicast forwarding entry or information about the base source for the multicast group g, it simply discards the message.

It is important to note that less control messages are required to update base source information in ASCORT when compared to PIM-SM. In PIM-SM, BSR broadcasts a bootstrap message periodically to provide updated RP information to all routers in the network. Unlike PIM-SM, the core multicasts an *UPDATE*( ) message periodically to provide up-to-date base source information to all first-hop routers in ASCORT.

3.3.7 Adaptability to Multicast Group Dynamics

The way how ASCORT deals with the case that a new source joins a multicast group is already given in Section 3.3.4.2. In this section, we provide the details how ASCORT deals with the cases that an existing source leaves its multicast group, a new member joins a multicast group and an existing member leaves its multicast group.

3.3.7.1 Existing Sources Becoming Inactive

When a source, s_i, leaves a multicast group, g, its first-hop router, fr_i, performs the following procedures.

1. If fr_i is not the core, that is, $fr_i \neq fr_b$, the timer associated with the interface connecting s_i in the incoming interface list of the wildcard multicast forwarding entry will expire and the interface will be deleted. This results in the expiration of the timers associated with the incoming interfaces in the multicast forwarding entries maintained at the member routers which have chosen s_i as their selected sources. As a result, the reduced tree rooted at s_i is destroyed. When the timers expire, these member routers send *GRAFT*(s_b, g) messages toward the base source s_b so as to construct a new branch in the reduced tree rooted at s_b.

2. If fr_i is the core, that is, $fr_i = fr_b$, it broadcasts a *CLEAR*(g) message. Upon receiving the *CLEAR*(g) message via interface $\{v\}$, a router deletes the base source information and the multicast forwarding entry for the multicast group g regardless of whether it is source-specific or wildcard. After the deletion of the base source information and the multicast forwarding

entry, the router forwards the *CLEAR*(g) message downstream via all interfaces except $\{v\}$. This makes the multicast distribution tree built by ASCORT to be destroyed.

When a source sends any packet to the multicast group g after the destruction of the multicast distribution tree, ASCORT considers the source as a new source joins and constructs the multicast distribution tree once again. Let us note that it is possible that a router will receive more than one *CLEAR*(g) messages. In case that the router has deleted the base source information and the forwarding entry for the multicast group g, it simply discards such messages. The details of the *UPDATE*(s_b, g) and the *CLEAR*(g) messages have been given in Section 3.3.6.3.

It is important to note that ASCORT is suitable for multicast applications in which the sources join a multicast group for a long period of time. Since ASCORT destroys the multicast distribution tree when a base sources departs, control messages are generated repeatedly if every base source joins the multicast group for a short period of time. For those applications in which the sources join a multicast group for a short period of time only, we can construct a bi-directional tree rooted at a pre-selected core. The multicast distribution tree rooted at a pre-selected core will not destroyed unless the core fails. Similar to PIM-SM, a pre-selected core cannot adapt to the location of sources and therefore longer average end-to-end delay may be resulted.

3.3.7.2 Hosts Joining the Multicast Group

When a host m_j wants to join the multicast group g, it uses IGMP to communicate with its LAN's designated router. If the designated router is not an on-tree router for g, it becomes a member router mr_j for g and sends a *GRAFT*(s_b, g) message toward the base source s_b. Furthermore, mr_j creates a source-specific multicast forwarding entry, (s_b, g, in_b, out_b) where in_b is the interface to which the *GRAFT*( ) message is sent and out_b contains the interface connecting m_j. Upon receiving the *GRAFT*(s_b, g) message via interface $\{v\}$, a router performs the grafting procedures mentioned in Section 3.3.6.1.

3.3.7.3 Hosts Leaving the Multicast Group

An on-tree router deletes the interface from the outgoing interface list in the source-specific or wildcard multicast forwarding entry when no host in its LAN has membership in the multicast group g.

1. If the router maintains a wildcard multicast forwarding entry, (*, g, *in*, *out*), and $out = \phi$, it deletes the forwarding entry.

2. If the router maintains a source-specific multicast forwarding entry, (s_i, g, in_i, out_i), and $out_i = \phi$, it sends a *PRUNE*(s_i, g) message upstream toward s_i. After sending the *PRUNE*(s_i, g) message, the router deletes this source-specific multicast forwarding entry. Upon receiving a *PRUNE*(s_i, g) message via interface $\{v\}$, a router performs the following procedures.

   2.1 If the router does not maintain any multicast forwarding entry for the multicast group g, it simply discards the *PRUNE*( ) message.

   2.2 Else if the router already maintains a wildcard multicast forwarding entry (*, g, *in*, *out*), it removes $\{v\}$ from *out* in condition that $\{v\} \in out$. If the removal of $\{v\}$ from *out* makes *out* become empty, the wildcard multicast forwarding entry will be deleted.

   2.3 Else if the router already maintains a source-specific multicast forwarding entry, (s_i, g, in_i, out_i), it removes $\{v\}$ from out_i in condition that $\{v\} \in out_i$. If the removal of $\{v\}$ from out_i makes out_i become empty, the router forwards the *PRUNE*(s_i, g) message upstream to in_i and then deletes the source-specific multicast forwarding entry.

The operation performed after receiving a *PRUNE*( ) message in ASCORT is different from that in PIM-DM. In PIM-DM, each outgoing interface is associated with a timer. When a router receives a prune message from an interface, the timer associated with the interface is reset. No multicast packet can be sent from this interface before the timer expires. In other words, when the timer timeouts, multicast packets are sent from this interface again. Unlike PIM-DM, ASCORT deletes the interface, from which the prune message is received, explicitly from the outgoing interface list of the source-specific multicast forwarding entry.

3.3.8 Adaptability to Network Dynamics

ASCORT makes use of soft-state mechanism to adapt to underlying network conditions. The ways to deal with the link failures on reduced trees, those on bi-directional trees, the failures of first-hop routers, and the failures of the core are given in Section 3.3.8.1, 3.3.8.2, 3.3.8.3, and 3.3.8.4 respectively.

3.3.8.1 Link Failures on Reduced Trees

If there is a link failure on the reduced tree rooted at a source regardless of whether it is the base source or not, a subtree is disconnected from the reduced tree. The timers associated with the incoming interfaces in multicast forwarding entries maintained at member routers on the disconnected subtree will eventually expire. Upon timeout, each of the member routers sends a *GRAFT*(s_b, g) message toward the base source. On the receipt of a *GRAFT*(s_b, g) message, a router

performs the procedures mentioned in Section 3.3.6.1. As a result, the member routers are connected to the reduced tree rooted at the base source.

3.3.8.2 Link Failure on the Bi-Directional Tree

In ASCORT, the first-hop router of each source periodically sends a *GRAFT*(*, *g*) message toward the base source. On the receipt of *GRAFT*(*, *g*) message, a router performs the procedures mentioned in Section 3.3.4.2.1. As a result, an alternative bi-directional route will be found if there is a link failure on the bi-directional tree.

3.3.8.3 Failures of First-Hop Routers

If a first-hop router other than the core is failed, the timers associated with the incoming interfaces in multicast forwarding entries maintained at member routers on the reduced tree rooted at that router will eventually expire. This makes each of the member routers send a *GRAFT*(s_b, *g*) message toward the base source. On the receipt of a *GRAFT*(s_b, *g*) message, a router performs the procedures mentioned in Section 3.3.6.1. As a consequence, the member routers are connected to the reduced tree rooted at the base source.

3.3.8.4 Failure of the Core

If the core fr_b is failed, the first-hop routers will not receive an *UPDATE*(s_b, *g*) message. Each of these first-hop routers will send a *CLEAR*(*g*) message to all outgoing interfaces. The details of the *UPDATE*(s_b, *g*) and the *CLEAR*(*g*) messages have been given in Section 3.3.6.3.

3.3.9 The Use of Timers

To adapt to group dynamics and the network dynamics, ASCORT employs soft-state mechanism. Each incoming interface in a multicast forwarding entry, including source-specific and wildcard multicast forwarding entry, is associated with a timer. The timer of an incoming interface is reset when (i) a router, which maintains a source-specific multicast forwarding entry (s_i, *g*, in_i, out_i), receives a multicast packet from the source s_i at that interface; or (ii) a router, which maintains a wildcard multicast forwarding entry (*, *g*, *in*, *out*), receives a *GRAFT*(*, *g*) message at that interface. The timer of an incoming interface in_i in a source-specific multicast forwarding entry expires implies s_i has left the multicast group *g* and/ or the first-hop router of s_i is failed. On the other hand, the timer of an incoming interface *in* in a wildcard multicast forwarding entry expires implies one of the sources has left the multicast group *g* and/ or the first-hop router of that source is failed.

If the router is the core, the expiration of the timer associated with the incoming interface indicates that the source in its LAN has left the multicast group. In this case, the router does not

send the *UPDATE*(s_b, g) messages. This results in the destruction of all reduced trees and the bi-directional tree. This is followed by the re-construction of multicast distribution tree.

Similarly, if the router is not the core, the expiration of the timer associated with the incoming interface indicates that the source in its LAN has become inactive in the multicast group g. The router will not send the *GRAFT*(*, g) messages toward the base source. The branch on the bi-directional tree connecting the router to the base source will be destroyed.

In case that the router is a member router, expiration of the timer associated with the incoming interface indicates that the root of the reduced tree it lies on has left the multicast group. The router will send a *GRAFT*(s_b, g) message toward the base source so as to build a tree branch on the reduced tree rooted at the base source.

Similar to the incoming interface, each interface in the outgoing interface list in a multicast forwarding entry is associated with a timer. The timer of an outgoing interface is reset when a *GRAFT*(s_i, g) or *GRAFT*(*, g) message is received at that interface. When the timer associated with an outgoing interface expires, the outgoing interface will be deleted. If the deletion of the outgoing interface results in an empty outgoing interface list, the forwarding entry will be deleted. In this case, if the router is a member router, it will send a *GRAFT*(s_b, g) message toward the base source so as to build a new tree branch on the reduced tree rooted at the base source.

The value of the timer can be set to a fixed value just as what PIM-SM does. As an alternative, the value of the timer can be set to a variable value so as to adapt to the volume of control traffic and available bandwidth on the network link [Sharma *et al.* 1997].

3.4 Properties of ASCORT

In this section, we describe the properties of ASCORT. We show that ASCORT can always build a multicast distribution tree given a connected network in which there exists some physical path between any pair of routers that can become part of the multicast distribution tree (Property 1). Furthermore, we show that the reduced trees constructed by ASCORT will not overlapped with one another (Property 2) and hence only one multicast forwarding entry is maintained in the routers lying on the multicast distribution tree constructed by ASCORT (Property 3). Moreover, we show that ASCORT is free of any routing loop in the multicast distribution tree (Property 4).

Property 1: A multicast distribution tree can always be built given a connected network.

Proof: The construction of a multicast distribution tree depends on the successful trigger of the protocol when new members and new sources join the multicast group.

Case 1: When a new member joins the multicast group, its designated router will become a member router. The member router will send a graft message

toward the core if it is not already on-tree. This resulted in the construction of a multicast distribution tree.

Case 2: When a new source joins the multicast group, it will broadcast its first multicast packet to the multicast group. Upon receiving such packet, the member router will select between this new source and the root of its reduced tree. In case that some member routers select the new source to join, a reduced tree rooted at the new source will be built. Furthermore, the first-hop router of the new source will also join the bi-directional tree. This constructs a multicast distribution tree.

As a result, a multicast distribution tree can always be built given a connected network.

Property 2: Reduced trees do not overlap with each other.

Proof: Let us consider the case that there are two member routers MR_i and MR_j and two sources, S_i and S_j such that the IP address of S_i is smaller than that of S_j. We suppose that MR_i and MR_j select S_i and S_j as its selected source, respectively. We further suppose that the branch connecting MR_i to S_i crosses over the branch connecting MR_j to S_j at a router R (i.e. the reduced tree rooted at S_i overlaps with the reduced tree rooted at S_j at a router R). We denote the minimum number of hops from a router X to another router Y as $H_{X,Y}$. Fig. 3.5 shows an example illustrating the scenario.

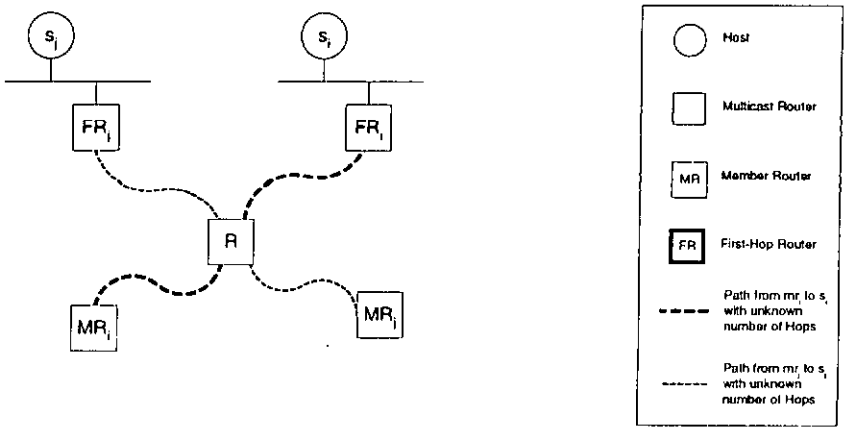


Fig. 3.5. An example illustrating how reduced trees are overlapped with each other.

Since MR_i selects S_i as its selected source, $H_{MR_i-S_i} \leq H_{MR_i-S_j}$. This implies, $H_{MR_i-R} + H_{R-S_i} \leq H_{MR_i-S_j}$. Thus $H_{MR_i-R} + H_{R-S_i} \leq H_{MR_i-S_j} \leq H_{MR_i-R} + H_{R-S_j}$ because there must exist a path from MR_i to R and from R to S_j. Consequently, $H_{R-S_i} \leq H_{R-S_j}$.

Similarly, since MR_j selects S_j as its selected source, $H_{MR_j-S_j} < H_{MR_j-S_i}$. Thus, $H_{MR_j-R} + H_{R-S_j} < H_{MR_j-S_i}$. Hence $H_{MR_j-R} + H_{R-S_j} < H_{MR_j-S_i} \leq H_{MR_j-R} + H_{R-S_i}$ because there must exist a path from MR_j to R and from R to S_i. Consequently, $H_{R-S_j} < H_{R-S_i}$.

The above introduces a contradiction. This implies that it is impossible for two reduced trees to overlap with one another.

Property 3: Only one multicast forwarding entry is maintained in the routers lying on the multicast distribution tree constructed by ASCORT.

Proof: Construction of multicast distribution tree in ASCORT involves two parts: construction of reduced tree and construction of the bi-directional tree. In the construction of reduced trees, each member router selects a source to join based on hop count. As proved in property 2, there is no overlapping between reduced trees. As a result, each router on the reduced tree maintains a source-specific multicast forwarding entry only. To build a bi-directional route, a first-hop router either replaces a source-specific multicast forwarding entry by a new wildcard multicast forwarding entry or modifies an existing wildcard multicast forwarding entry and sends a *GRAFT*(*, g) message to a core. On the receipt of a *GRAFT*(*, g) message, a router creates a new wildcard multicast forwarding entry if it does not maintain any multicast forwarding entry. If a router already maintains a wildcard multicast forwarding entry, it modifies the entry. Otherwise, if a router already maintains a source-specific multicast forwarding entry, it replaces the entry by a new wildcard multicast forwarding entry. As a result, each router lying on the bi-directional route maintains a wildcard multicast forwarding entry. Consequently, only one multicast forwarding entry is maintained in the routers lying on the multicast distribution tree constructed by ASCORT.

Property 4: No loop can be formed with on-tree routers when the underlying unicast routing protocol is stable and does not contain any loop.

Proof: Assume that the underlying unicast routing protocol is stable and does not contain any loop, and a loop has formed in the multicast distribution tree built by ASCORT. This loop must involve two on-tree routers, R_1 and R_2, such that the path from R_1 to R_2 passes through itself.

Case 1: Both R_1 and R_2 lie on the same reduced tree. Since the path between R_1 and R_2 is the shortest path, the scenario that the shortest path from one router to another router passes through itself violates the assumption that the underlying unicast routing protocol is stable and loop free. As a result, such a loop cannot be formed.

Case 2: Both R_1 and R_2 lies on the bi-directional tree. A graft message is forwarded from a first-hop router to the core using the shortest paths in the bi-directional tree. It is impossible that a path passes through the same router more than once. Consequently, the path from R_1 to R_2 cannot pass through itself and hence no loop can be formed.

Case 3: R_1 lies on a reduced tree, T_1, and R_2 lies on the bi-directional tree. Since the root of a reduced tree is connected to the bi-directional tree, a loop may be formed on the path from R_1 to the root of T_1 or on the path from the root of T_1 to R_2. However, such loop cannot be formed as shown in Case 1 and Case 2. As a result, it is impossible to form any loop in this case.

Case 4: R_1 lies on one reduced tree, T_1, and R_2 lies on another reduced tree, T_2. Since the root of a reduced tree is connected to the bi-directional tree, there must be a loop on the path from the root of T_1 to the root of T_2 to form a loop. However, this loop is impossible to be formed as shown in Case 2. It is for this reason that no loop can be formed in this case.

As a result, no loop can be formed when the underlying unicast routing protocol is stable and does not contain any loop.

3.5 Simulation Results

In our experiments, we represent network topologies in the form of random graphs. These random graphs are generated using the algorithms described in [Waxman 1988]. A random graph, N, is denoted as $N = (V, E)$ where V is a set of nodes which represents multicast routers and E is a set of

edges which represents network links. These nodes are randomly distributed over a Cartesian coordinate system. The probability that an edge exists between any two nodes u and v is given by the probability function, $P(\{u, v\})$, which is, in turn, defined as

$$P(\{u,v\}) = \beta \times e^{\frac{-d(u,v)}{L\alpha}} \tag{3.1}$$

where $d(u, v)$ is the distance from u to v, L is the maximum distance between the two nodes, and α and β are parameters in the range (0, 1]. The density of short edges relative to longer ones decreases as α increases whereas the edge densities increases as β increases [Waxman 1988].

In our simulation experiments, we set the average nodal degree to four and the cost of each link is assumed to be unity. We then generate a random graph consists of 100 nodes. We evaluated our multicast routing protocols under two different *group densities*, 0.05 and 0.2, where group density (GD) of a multicast group is defined as the fraction of member routers in the network. We also vary the number of first-hop router from 5 to 95. We assume a first-hop router has a host in its LAN sending multicast packets to a given multicast group (i.e. a first-hop router has a source in its LAN) whereas a member router has a host in its LAN that wants to receive multicast packets for a given multicast group (i.e. a member router has a group member in its LAN). The number of sources and that of members are therefore the same as the number of first-hop routers and that of member routers, respectively. Let us note that each data point in our simulation results is an average over 100 independent simulation experiments. In addition to the averages, we also measure the 95% confidence interval of each data point. The 95% confidence interval is shown in the form of error bar for each data point in the figures.

To allow detailed study of the performance of ASCORT, we consider different kinds of distribution of first-hop routers and member routers in the Internet. In random distribution of first-hop routers/member routers, each node in the random graph has equal probability to be a first-hop router/member router. On the other hand, in localized distribution of first-hop routers/member routers, we select a node in the random graph to be a first-hop router/member member in a random manner. We then choose the neighbors around this node to be first-hop routers/member routers. The neighbors around these first-hop routers/member routers are selected to be first-hop routers/member routers and so on. Such process repeats until the required number of first-hop routers/member routers has been reached. This makes first-hop routers/member routers clustered in an area in the random graph. Let us note that a router may be selected as a first-hop router and a member router at the same time.

In our simulations, we consider four different scenarios. Firstly, we consider random distribution of first-hop routers and member routers (Scenario 1). Secondly, we consider random distribution of first-hop routers and localized distribution of member routers (Scenario 2). Thirdly,

we consider localized distribution of first-hop routers but random distribution of member routers (Scenario 3). Lastly, we consider localized distribution of first-hop routers and member routers (Scenario 4).

We evaluate the effect of ASCORT in the areas of multicast states, packet delay in terms of hop counts, and overhead of control messages. Furthermore, we also apply PIM-DM and PIM-SM to facilitate comparisons. We select rendezvous point (RP) in PIM-SM randomly from the graph. In PIM-SM, RP connects to each first-hop router using source-specific path or tunnel. We denote the method of using source-specific paths and tunnels as PIM-SM-s and PIM-SM-t respectively.

3.5.1 Scalability Analysis

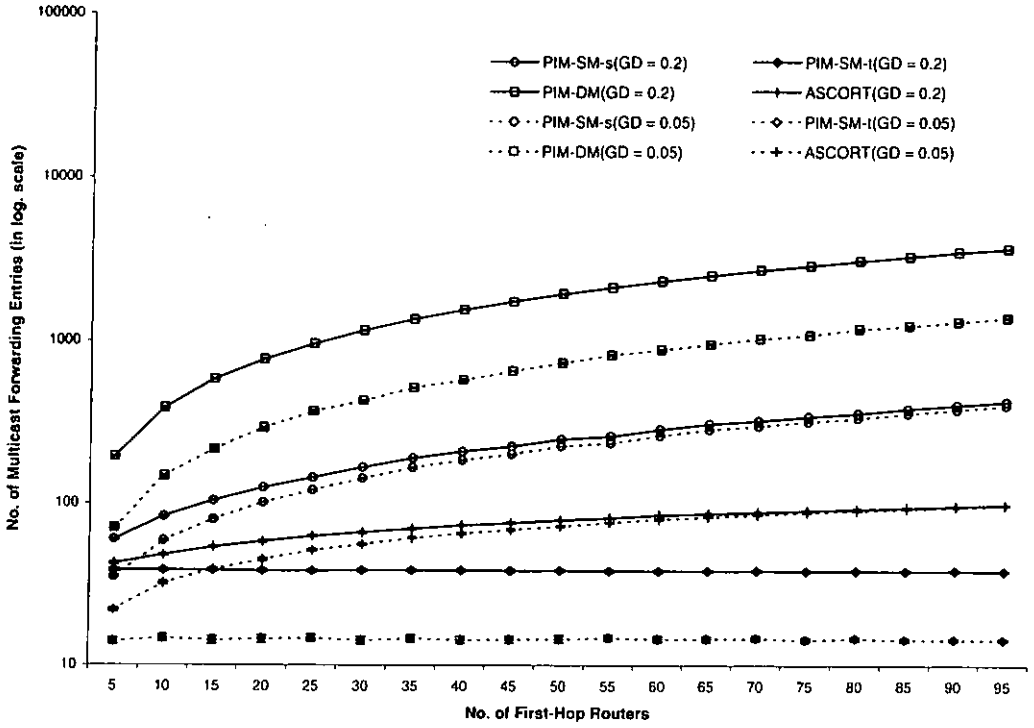
Different protocols have different requirements in implementing a multicast forwarding entry. A multicast forwarding entry, in general, contains four pieces of information: a source address, a multicast group address, an incoming interface set, and a set of outgoing interfaces. Nevertheless, some multicast routing protocols contain more information, for example, the soft-state timer value and the RP-bit in PIM-SM. Consequently, different multicast routing protocols use different number of bits to implement their own multicast forwarding entries. For simplicity, we assume the number of bits used to implement a multicast forwarding entry to be the same in PIM-DM, PIM-SM-s, PIM-SM-t, and ASCORT.

Based on such assumption, we measure the scalability of a multicast routing protocol in terms of the number of multicast forwarding entries required at routers to facilitate multicast routing. Since all multicast routing protocols build multicast distribution trees for disseminating multicast packets, on-tree routers are required to maintain forwarding entries. As a result, the scalability of a multicast routing protocol is given by the number of multicast forwarding entries maintained at on-tree routers. It is possible that a multicast routing protocol builds more than one multicast distribution trees for a multicast group. The scalability is then calculated by the number of multicast forwarding entries maintained at routers lying on all the trees.

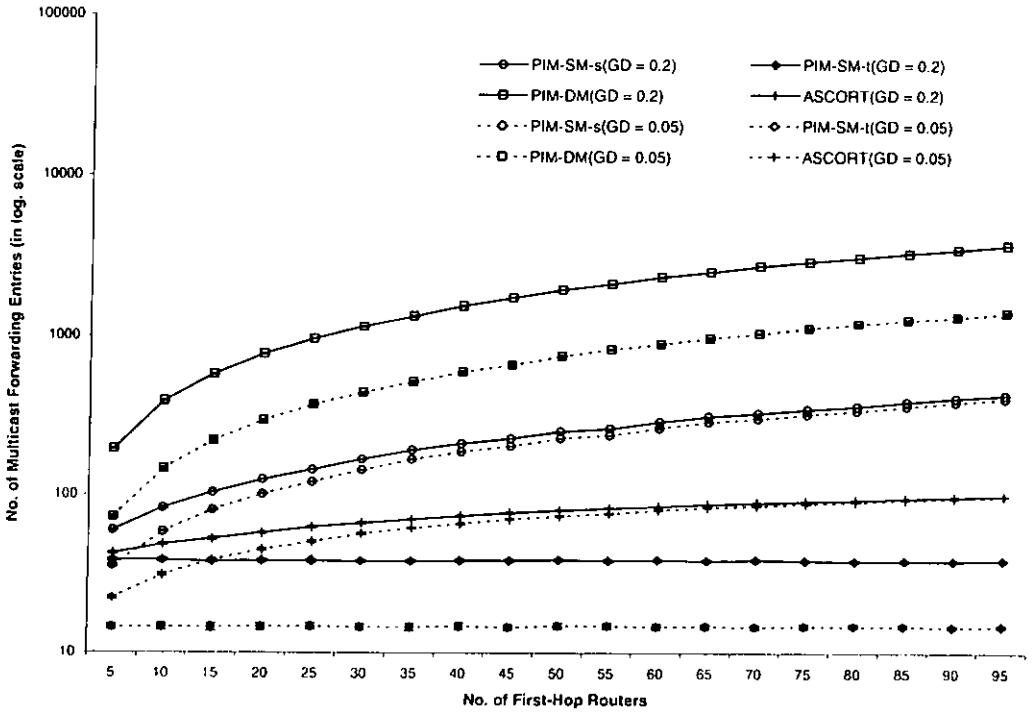
Given a random network, $N = (V, E)$, each router maintains a forwarding table which, in turn, stores a set of entries. It is these entries in the forwarding table allow multicast routing in the network. Taking a router, $v_i \in V$, into consideration, we denote the forwarding table maintained at v_i as T_i and the number of entries stored in T_i as $|T_i|$. The number of multicast forwarding entries maintained at multicast routers in a multicast routing protocol is defined as

$$\text{No. of multicast forwarding entries} = \sum_{\forall v_i \in V} |T_i| \tag{3.2}$$

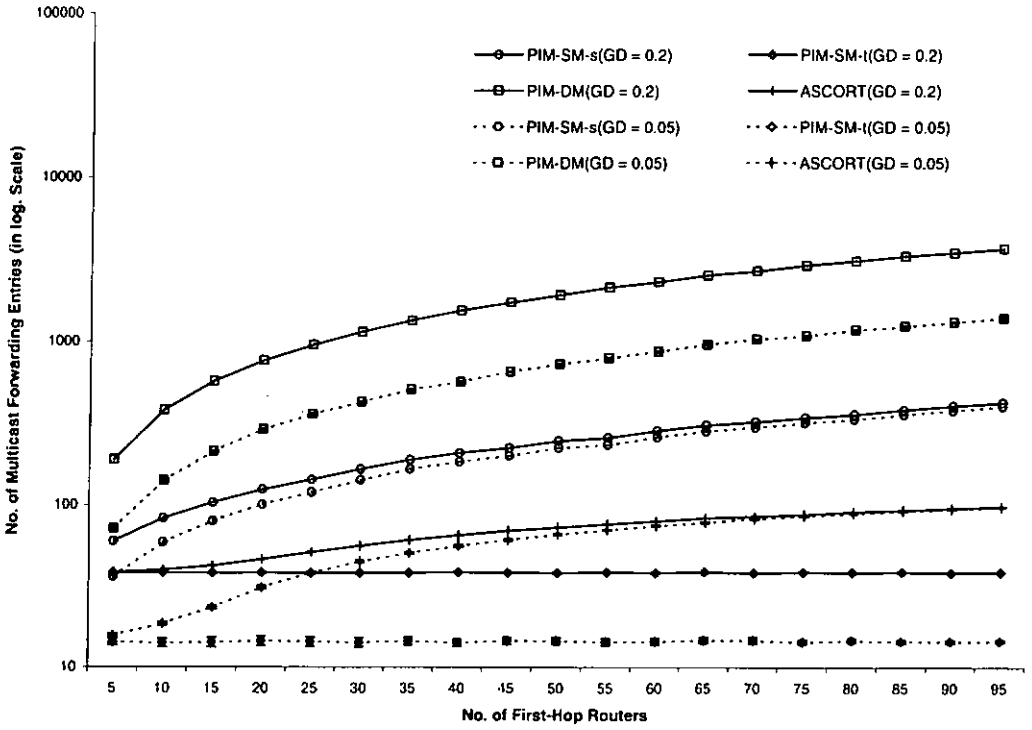
Fig. 3.6 shows the performances of PIM-DM, PIM-SM, and ASCORT in different distributions of first-hop routers and member routers under group density of 0.05 and 0.2.



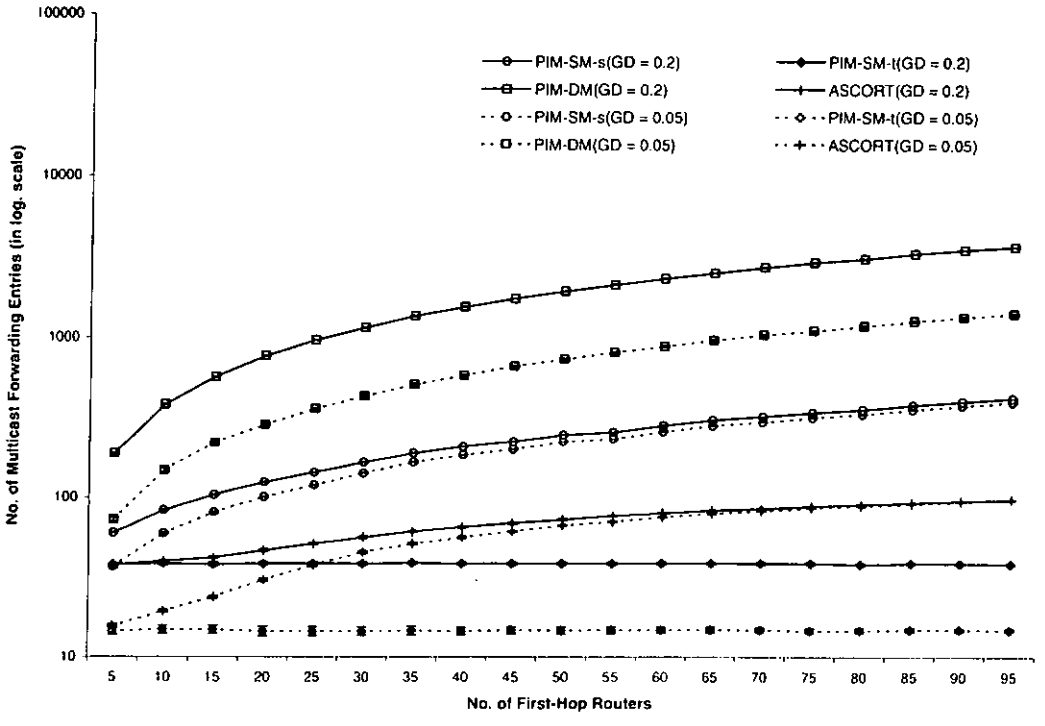
(a) Scenario 1



(b) Scenario 2



(c) Scenario 3



(d) Scenario 4

Fig. 3.6. No. of multicast forwarding entries maintained at routers in different scenarios.

As shown in Fig. 3.6, the performances of PIM-DM, PIM-SM-t, and PIM-SM-s do not exhibit any significant change in all the scenarios. This shows that the performances of these protocols in terms of number of multicast forwarding entries maintained at routers are independent of the distributions of first-hop routers and member routers. Multicast routing protocols based on source-specific trees such as PIM-DM build a multicast distribution tree for every source to connect to all group members. Since the location of a first-hop router is independent of the location of member routers, the number of on-tree routers is more or less the same in the source-specific tree and hence the number of multicast forwarding entries is nearly the same under different scenarios. As a result, the performance of PIM-DM is more or less the same no matter how the first-hop routers and member routers are distributed.

Unlike PIM-DM, multicast routing protocols based on shared trees such as PIM-SM, including PIM-SM-t and PIM-SM-s, build a shared tree rooted at a RP for delivering packets. Since the RP is selected from all the routers in the network randomly and the location of the RP is independent of the locations of first-hop routers and member routers, the number of on-tree routers does not change significantly under different distribution of first-hop routers and member routers. The number of multicast forwarding entries in the shared tree is therefore seemingly unchanged in different scenarios. It is for this reason that the performances of PIM-SM-t and PIM-SM-s are independent of the distributions of first-hop routers and member routers.

In addition to construct a shared tree rooted at RP connecting all member routers, a tree branch is built from the RP to each first-hop router in PIM-SM-s. Each router lying on the tree branches is required to maintain a source-specific multicast forwarding entry. On the other hand, a tunnel is built connecting each first-hop router to the RP in PIM-SM-t. Therefore, no source-specific multicast forwarding entry is required in PIM-SM-t. As a result, the number of forwarding entries in PIM-SM-s is more than that in PIM-SM-t. The number of forwarding entries maintained at the routers lying on the tree branches connecting first-hop routers to the RP gives the difference in scalability between PIM-SM-s and PIM-SM-t. Specifically, the number of entries in PIM-SM-t is independent of the number of first-hop routers because there is a single shared tree connecting member routers only. Nevertheless, the number of entries in PIM-SM-s increases as the number of first-hop routers increases because source-specific paths are constructed in addition to the shared tree.

On the other hand, ASCORT performs more or less the same in Scenario 1 and 2 and it performs in a similar manner in Scenario 3 and 4. This implies that the performance of ASCORT in terms of number of multicast forwarding entries maintained at routers are dependent on the distribution of first-hop routers but are independent of the distribution of member routers. In ASCORT, all the first-hop routers are connected by a bi-directional tree. When the first-hop routers are distributed locally, the size of the bi-directional tree is comparatively smaller than that

constructed under a random distribution of first-hop routers. In other words, the number of routers on the bi-directional tree constructed when the first-hop routers are distributed locally is smaller than that constructed when the first-hop routers are distributed randomly. It is for this reason that the performance of ASCORT in localized distribution of first-hop routers is better when compared to random distribution of first-hop routers. It is important to note that the performances of ASCORT in all the four scenarios become more or less the same as the number of first-hop routers increases. Such phenomenon is because of the fact that almost all of the routers in the network become first-hop routers and they are connected by a bi-directional tree in all the scenarios.

The number of multicast forwarding entries in ASCORT is significantly less than that in PIM-DM. It is because a single multicast distribution tree is constructed for a multicast group in ASCORT. Each router lying on the multicast distribution tree is required to maintain one multicast forwarding entry for a multicast group disregarding the number of sources in that multicast group. On the other hand, a separate source specific tree is constructed for each source in PIM-DM and hence a router may maintain several source specific entries for a multicast group.

When compared to PIM-SM-t, the number of forwarding entries in ASCORT is large in amount. Although both ASCORT and PIM-SM-t construct one multicast distribution tree for a multicast group, ASCORT constructs a distribution tree which connects all the first-hop routers and member routers together. Nevertheless, in PIM-SM-t, only member routers are connected together. It is for this reason that ASCORT maintains larger number of entries than PIM-SM-t does.

On the other hand, the number of forwarding entries in ASCORT is small in amount when compared to PIM-SM-s. Since a source-specific path is required to connect the RP to every first-hop router in PIM-SM-s, a forwarding entry is required to maintain at each router along the source-specific path. As a result, the number of forwarding entries in PIM-SM-s is larger than that in ASCORT.

3.5.2 Average End-to-End Delay

We measure the end-to-end delay from a source to a member in a multicast group in terms of hop counts. Given a random network, $N = (V, E)$, and a multicast group, g, let us suppose that there is a set of sources, S_g, and a set of members, M_g, in the multicast group g. The average end-to-end delay between each pair of sources and members is defined as

$$\text{avg. end - to - end delay} = \frac{\sum_{s \in S_g, m \in M_g} h_g(s, m)}{|S_g| \times |M_g|} \tag{3.3}$$

where $h_g(s, m)$ is the number of hop counts between s and m in the multicast distribution tree for g and $|\mathcal{S}|$ denotes the cardinality of a set, $\mathcal{S}$.

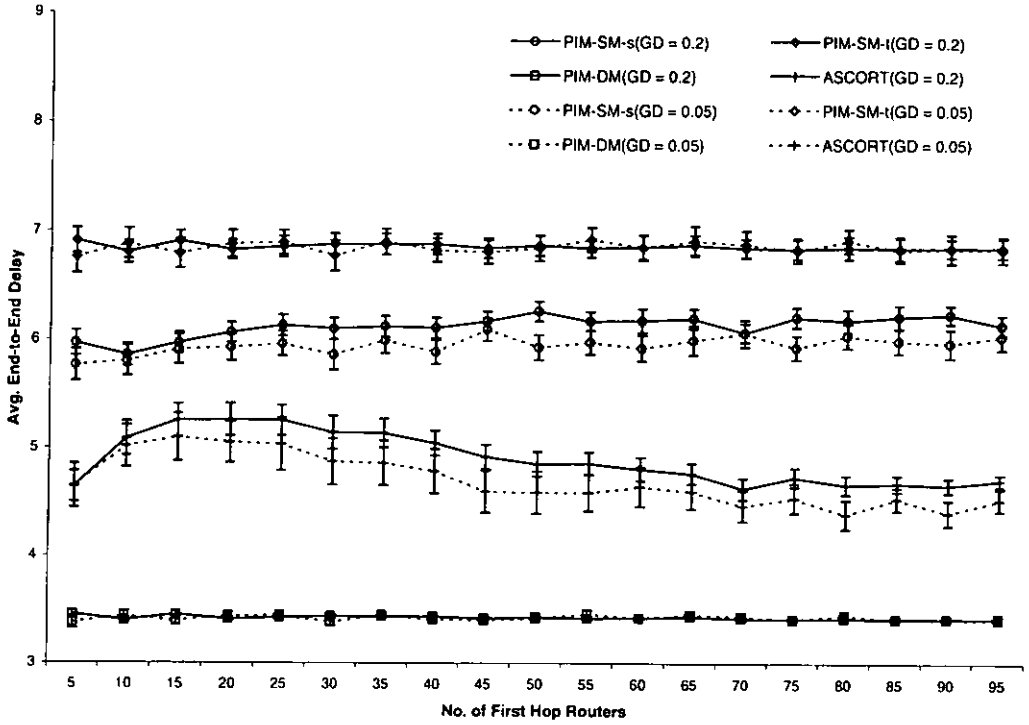
The average end-to-end delay of PIM-DM, PIM-SM-s, PIM-SM-t, and ASCORT in different distributions of first-hop routers and member routers under group density of 0.05 and 0.2 are given in Fig. 3.7. As shown in Fig. 3.7, the performances of PIM-DM, PIM-SM-s, and PIM-SM-t in terms of average end-to-end delay do not change significantly in all the four different scenarios. Multicast routing protocols based on source-specific trees such as PIM-DM build multicast distribution trees such that every pair of sources and members are connected by the shortest path. Consequently, PIM-DM is able to adapt to the locations of sources and members and exhibit optimal performances in different scenarios.

On the other hand, multicast routing protocols based on shared trees such as PIM-SM, including PIM-SM-s and PIM-SM-t, builds a shared tree rooted at a RP which is selected in a random manner. The shared tree is unable to adapt to the locations of sources and members. It is for this reason that the performances of PIM-SM-s and PIM-SM-t are worse than that of PIM-DM in different scenarios.

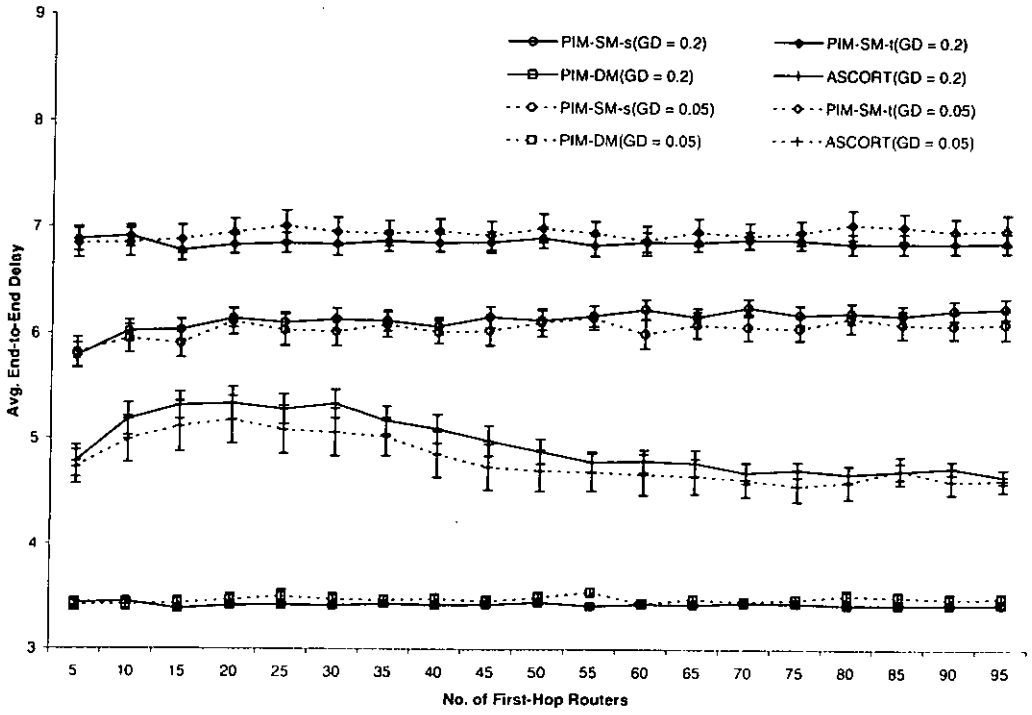
PIM-SM-s exhibits a more efficient packet distribution than PIM-SM-t does because of the construction of source-specific path from the RP to each first-hop router in PIM-SM-s. In PIM-SM-s, when source-specific path overlaps with the shared tree, multicast packets sent by a source may reach some group members without passing through the RP when these members are located at the subtree rooted at the overlapping nodes. However, in PIM-SM-t, multicast packets must be encapsulated by the first-hop routers and then unicast to the RP. On the receipt of the encapsulated packet, RP decapsulates the packet and then disseminates the multicast packet on the shared tree. Although the tunnel overlaps with the shared tree, multicast packets have to pass through the RP first before disseminating the multicast packets on the shared tree. As a result, the average end-to-end delay in PIM-SM-s is less than that in PIM-SM-t.

Unlike PIM-DM, PIM-SM-s, and PIM-SM-t, the performances of ASCORT are different in different scenarios. Specifically, the average end-to-end delay in ASCORT grows to a certain threshold and then shrinks as number of first-hop routers increases in Scenario 1 and 2 (Fig. 3.7 (a)-(b)) where first-hop routers are distributed in a random manner. For a source to send a multicast packet to all group members, it sends a copy to its reduced tree and a copy to the bi-directional tree. The packet traverses in the bi-directional tree reaches all the first-hop routers and is then delivered in the reduced trees so as to reach all group members. When the number of first-hop routers increases from 5 to 40, the average number of hops from a first-hop router to another first-hop router on the bi-directional tree increases. When the number of first-hop routers is greater than 40, the average number of hops from a first-hop router to another first-hop router on the bi-directional tree decreases. It is because of the fact that more and more routers, which already lie on the bi-directional tree, become first-hop routers when the number of first-hop routers

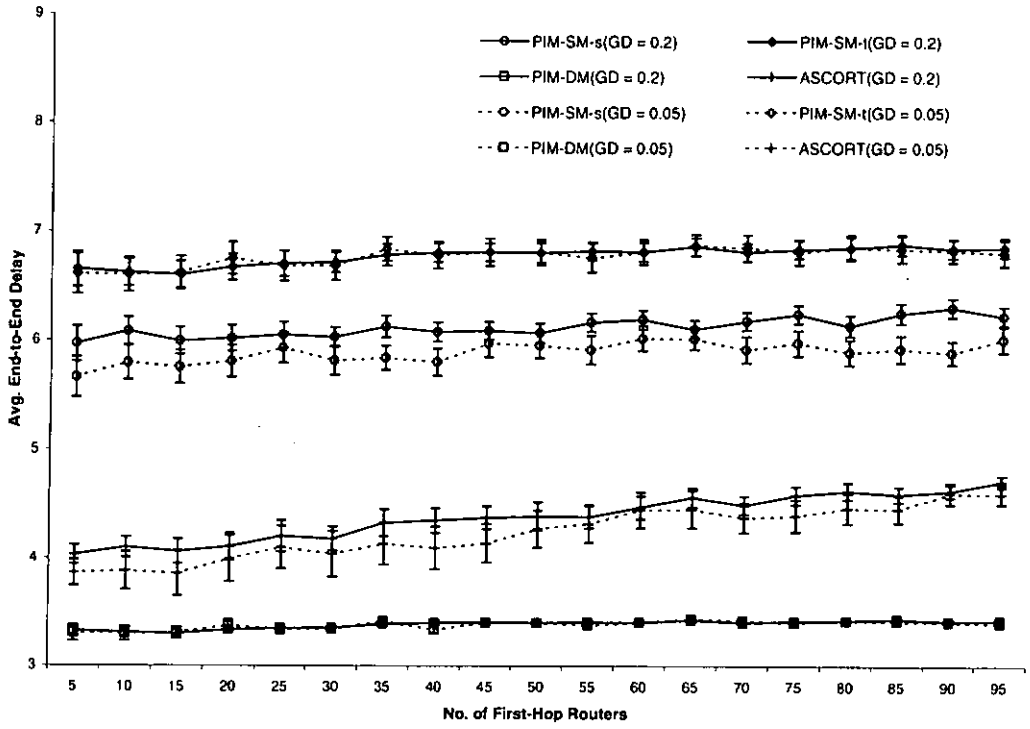
increases. It is for this reason that the average end-to-end delay increases to a certain value and then decreases with the number of first-hop routers (Fig. 3.7 (a)-(b)).



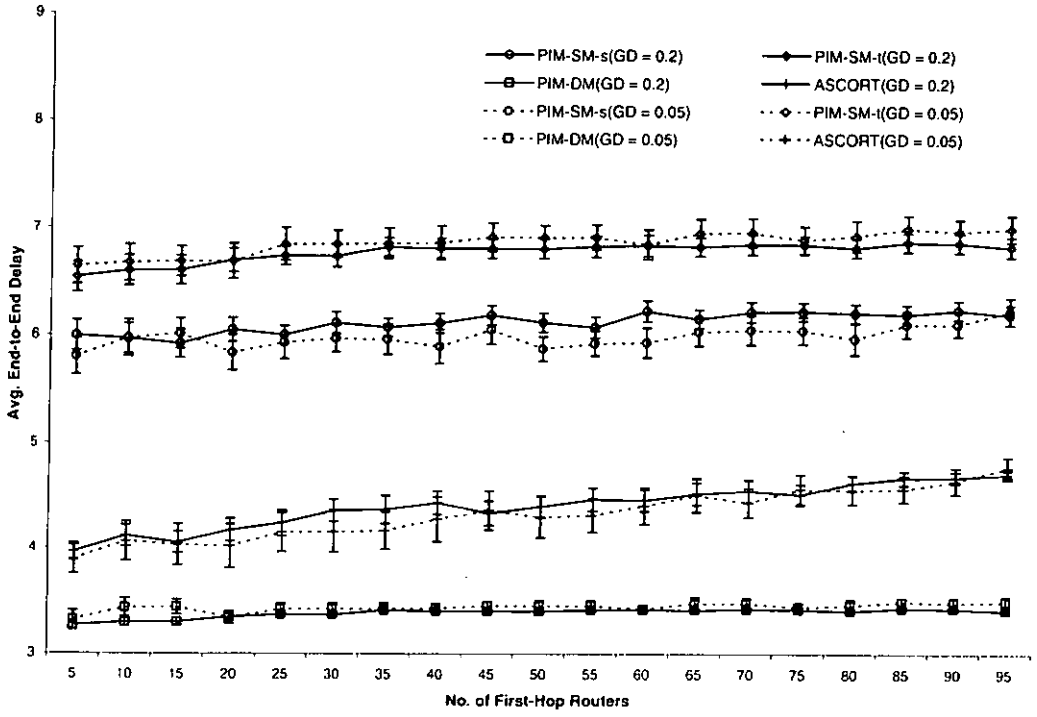
(a) Scenario 1



(b) Scenario 2



(c) Scenario 3



(d) Scenario 4

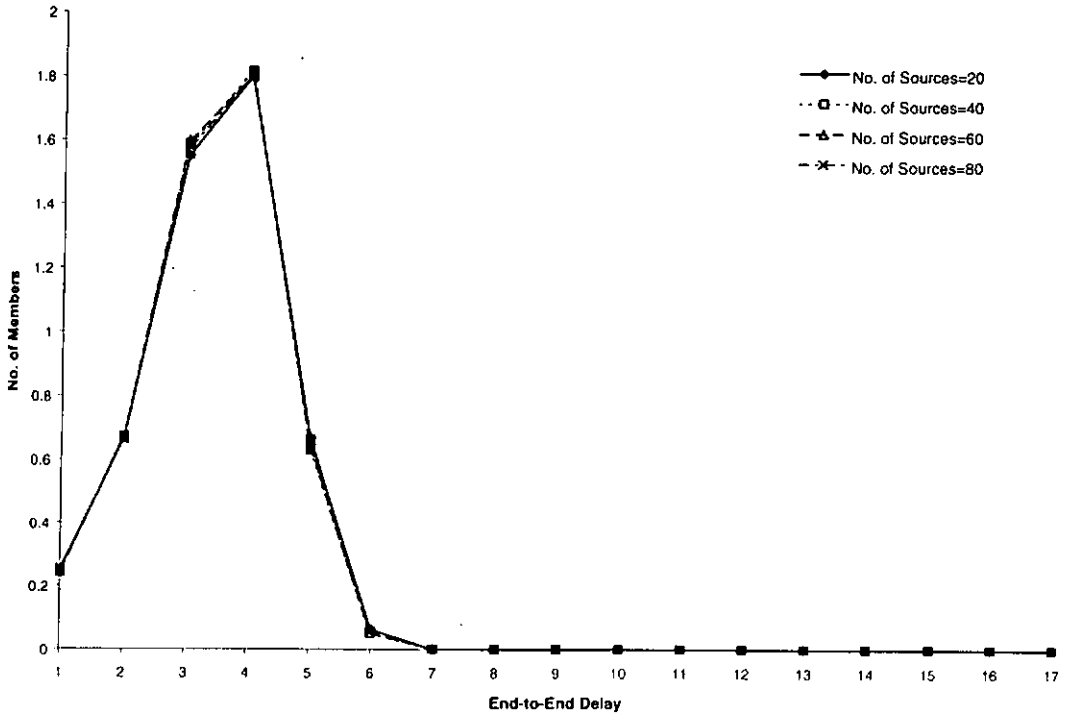
Fig. 3.7. Average end-to-end in different scenarios.

On the other hand, the average end-to-end delay in ASCORT increases as the number of first-hop routers increases in Scenario 3 and 4 (Fig. 3.7 (c)-(d)) where first-hop routers are distributed locally. It is due to the fact that the average number of hops from a first-hop router to another first-hop router on the bi-directional tree increases when the number of sources increases.

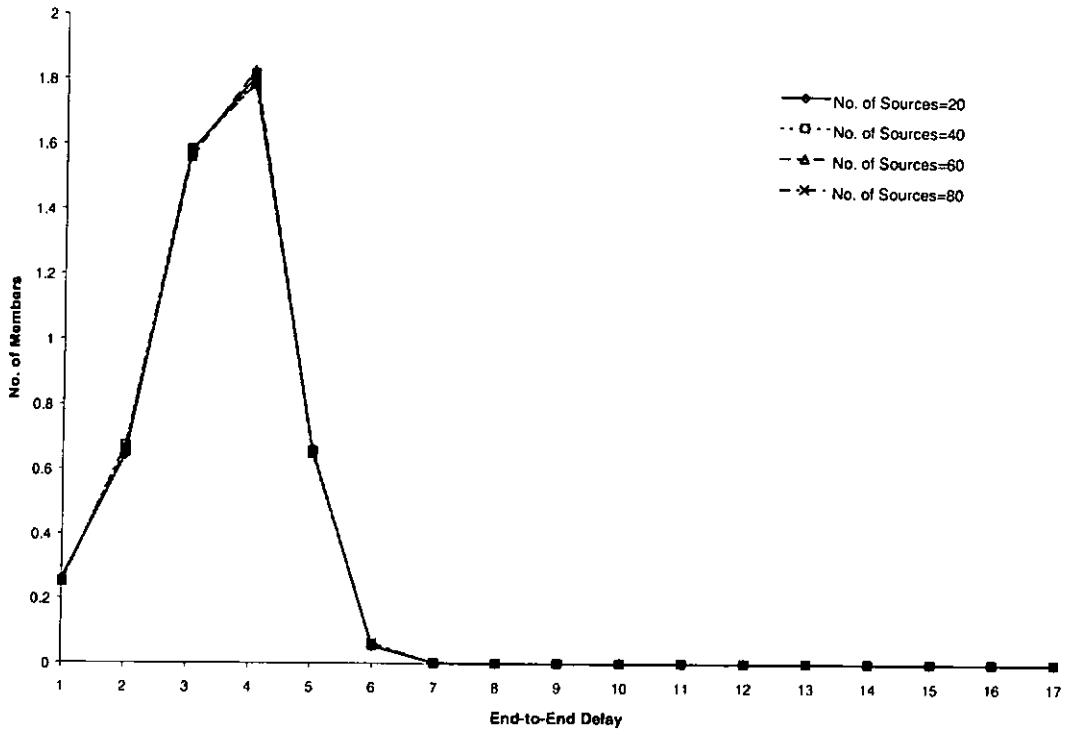
Let us note that ASCORT obtains 1.5-2 hops longer average end-to-end delay than that of PIM-DM in Scenario 1 and 2. Moreover, in Scenario 3 and 4, ASCORT obtains 1-1.5 hops longer average end-to-end delay than that of PIM-DM. On the other hand, PIM-SM-s obtains 2.5 hops and PIM-SM-t obtains 3.5 hops longer average end-to-end delay than that of PIM-DM in all the scenarios.

3.5.3 Delay Distribution

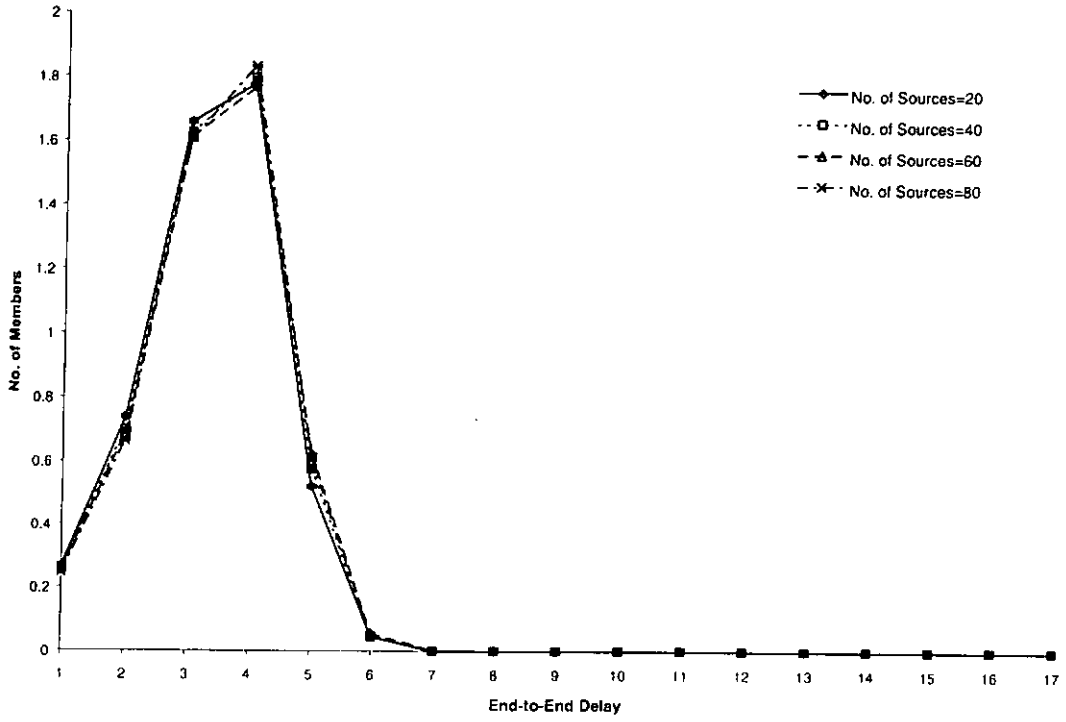
In addition to average end-to-end delay, we discuss how many group members receive multicast packets with different end-to-end delay. In this section, we present the distribution of members receiving packets with different end-to-end delay. The end-to-end delay distribution in PIM-DM with group density of 0.05 is shown in Fig. 3.8.



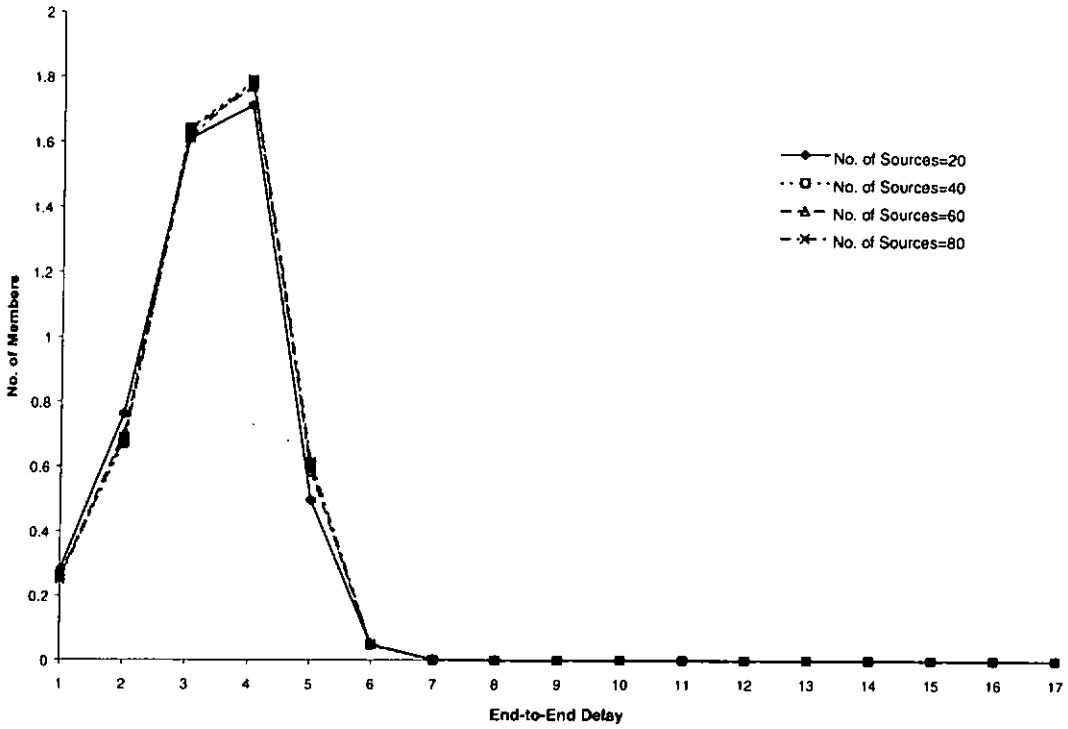
(a) Scenario 1



(b) Scenario 2



(c) Scenario 3

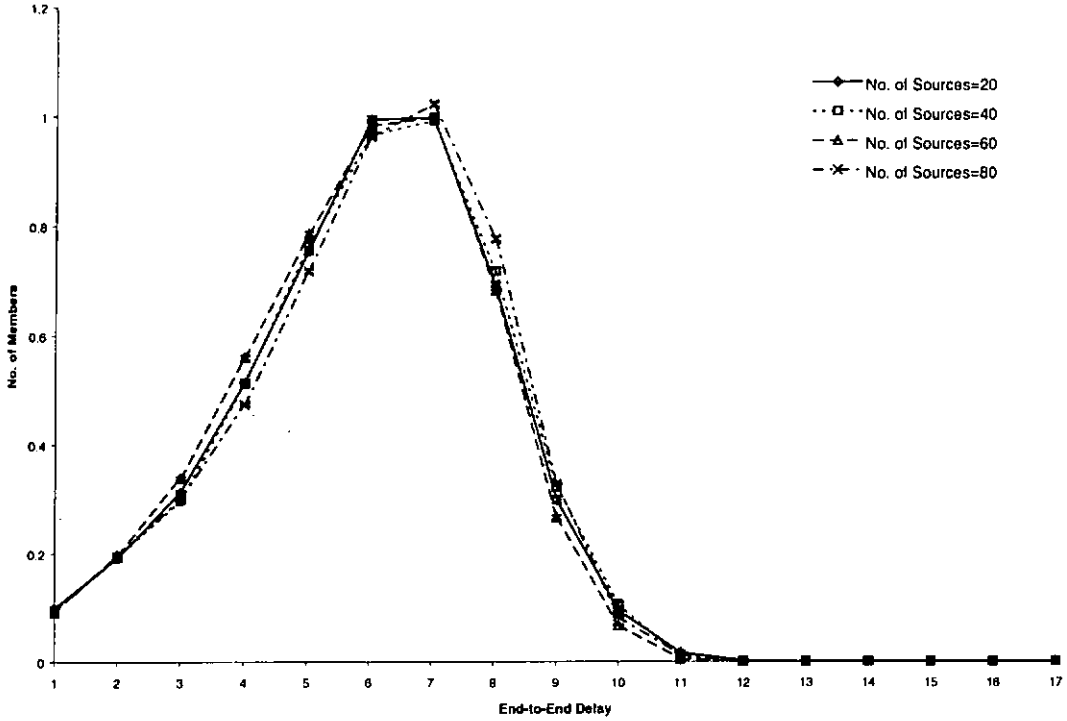


(d) Scenario 4

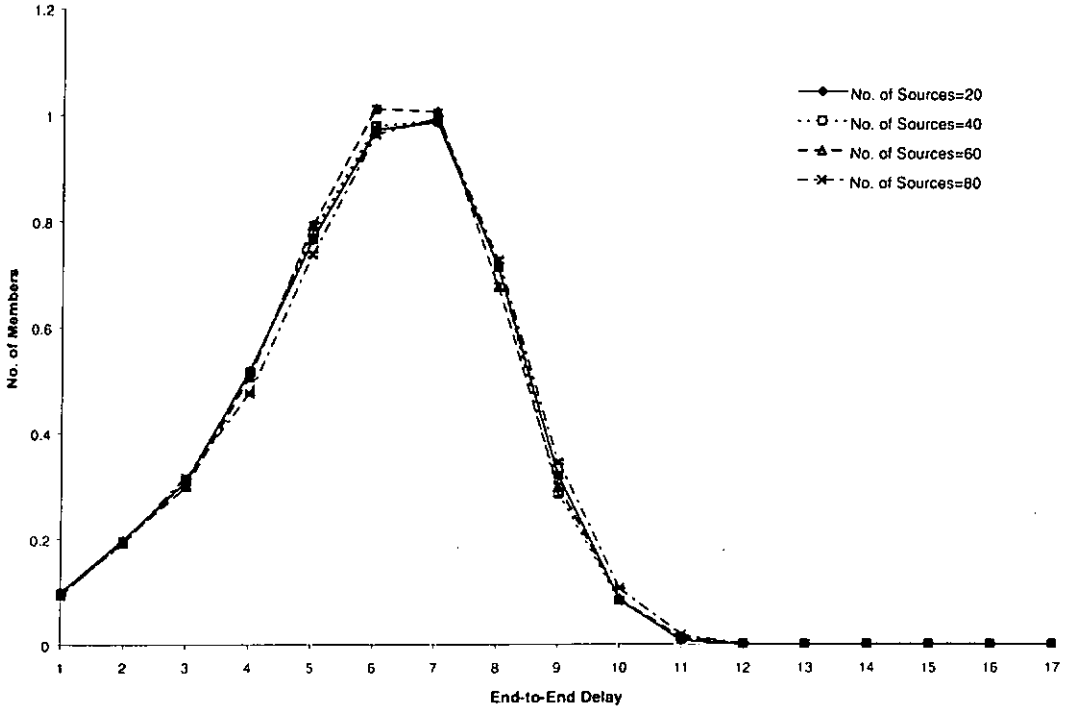
Fig. 3.8. End-to-end delay distribution of PIM-DM.

As shown in Fig. 3.8, the distributions of group members receiving multicast packets with different end-to-end delay are more or less the same with different number of sources in different scenarios. This results in more or less the same average end-to-end delay with different number of sources in different scenarios (Fig. 3.7). Each distribution exhibits a shape of a hill with the peak at 4 hops (Fig. 3.8). Most of group members receive multicast packets after 3-4 hops from a source. It is important to note that there is about 0.23 member out of 5 group members experiencing 1 hop end-to-end delay from a source.

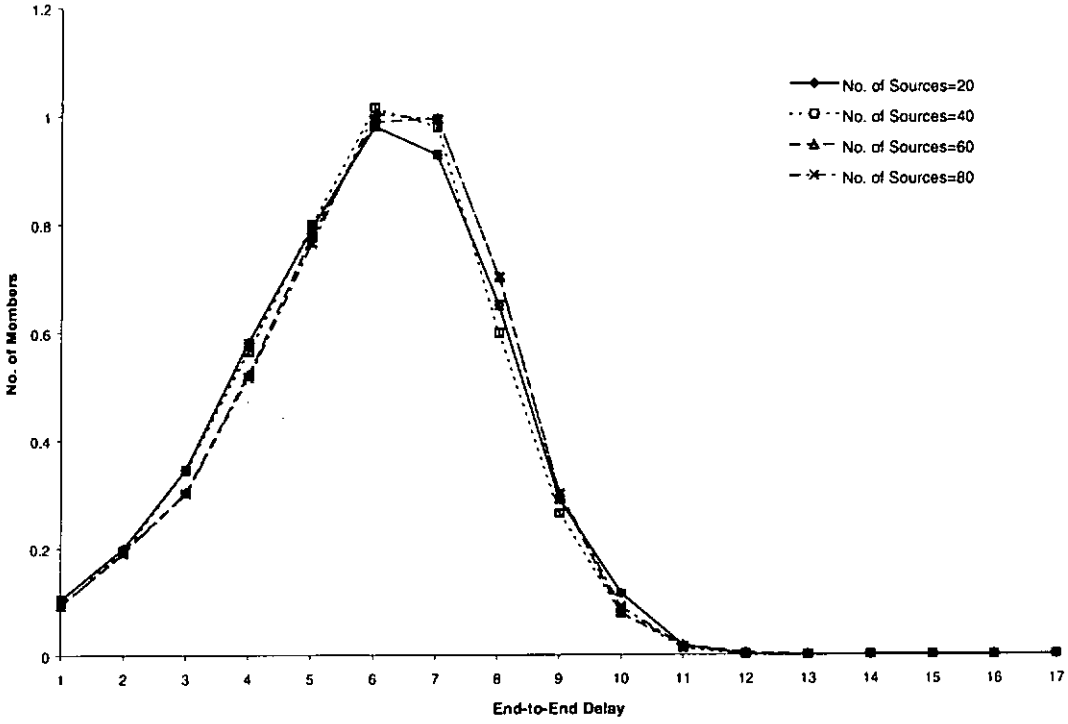
The end-to-end delay distribution in PIM-SM-s with group density of 0.05 is shown in Fig. 3.9. Similar to PIM-DM, the numbers of group members in PIM-SM-s experiencing different end-to-end delay are more or less the same with different number of sources in different scenarios. It is for this reason that the average end-to-end delays with different number of sources in different scenarios are more or less the same (Fig. 3.7). Each of the end-to-end delay distributions shows a shape of a hill with the peak at 6 hops (Fig. 3.9). Most of group members receive multicast packets from a source after 6-7 hops. It is less efficient than PIM-DM in which most members receive packets after 3-4 hops. There is about 0.1 member out of 5 group members experiencing 1 hop end-to-end delay from as source.



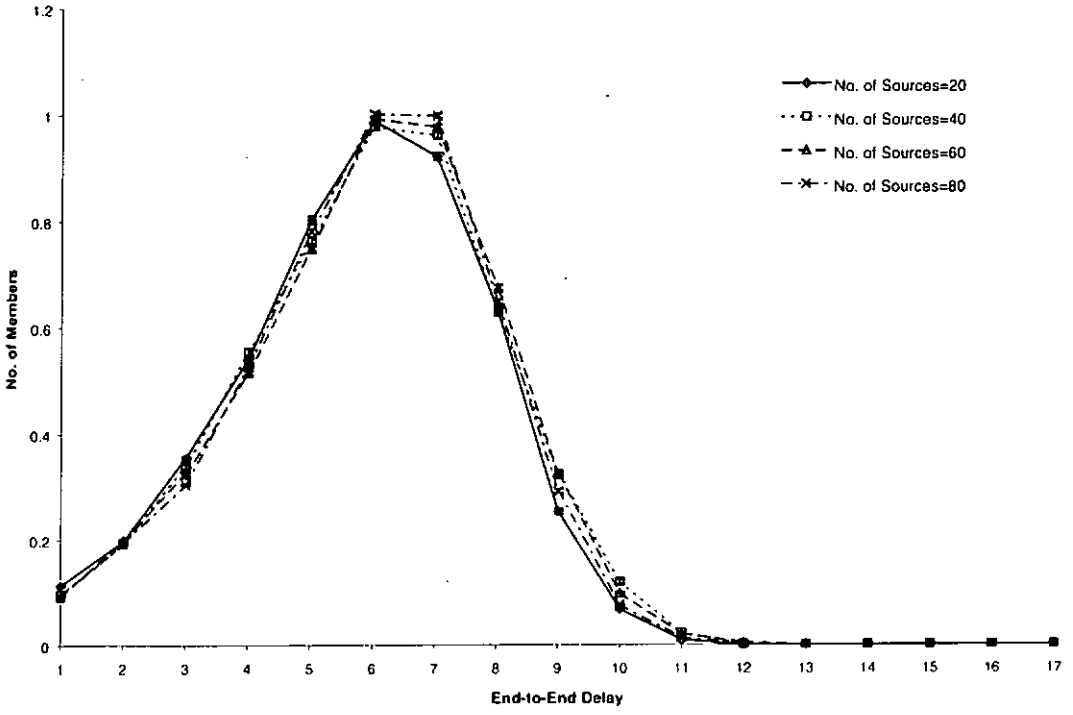
(a) Scenario 1



(b) Scenario 2



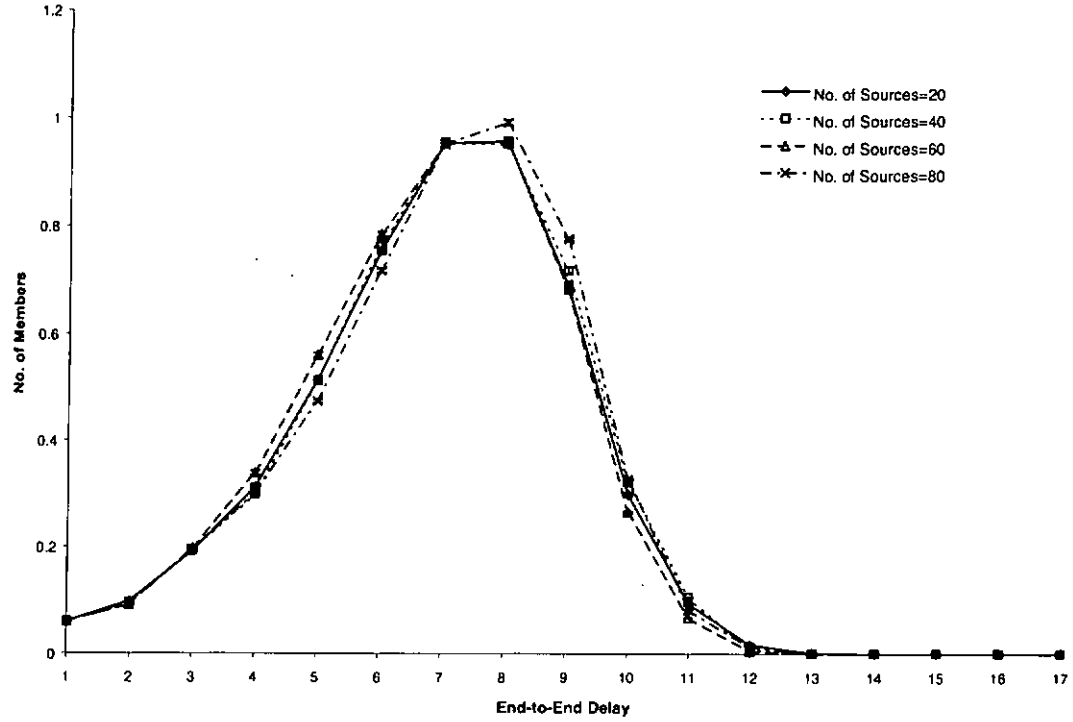
(b) Scenario 3



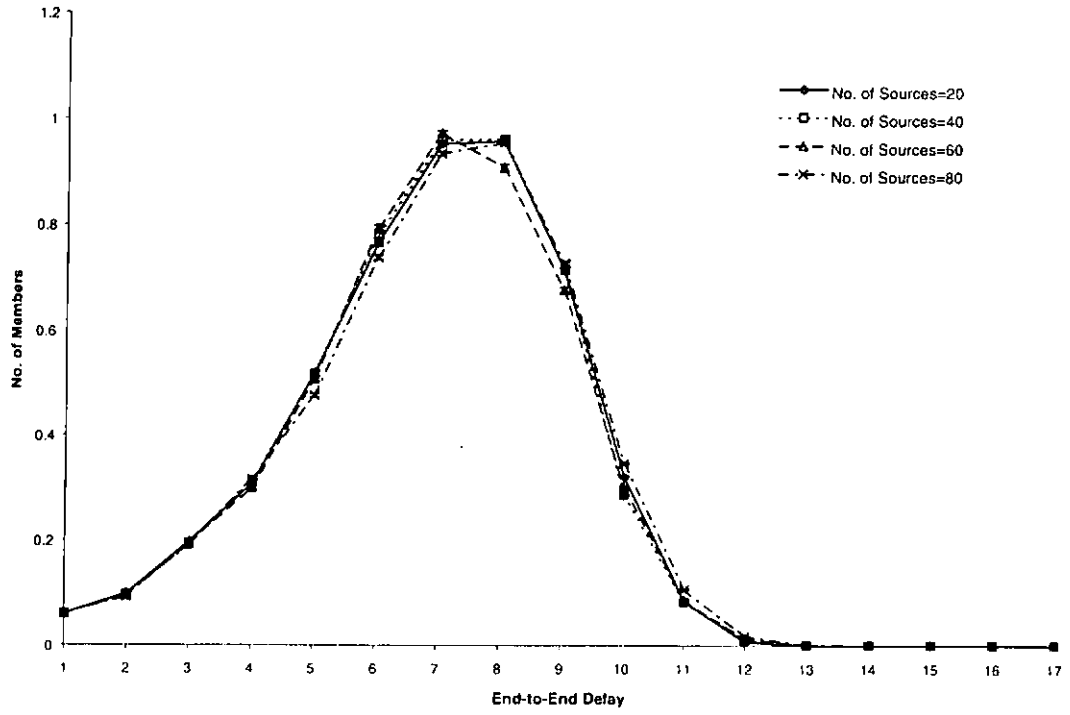
(d) Scenario 4

Fig. 3.9. End-to-end delay distribution of PIM-SM-s.

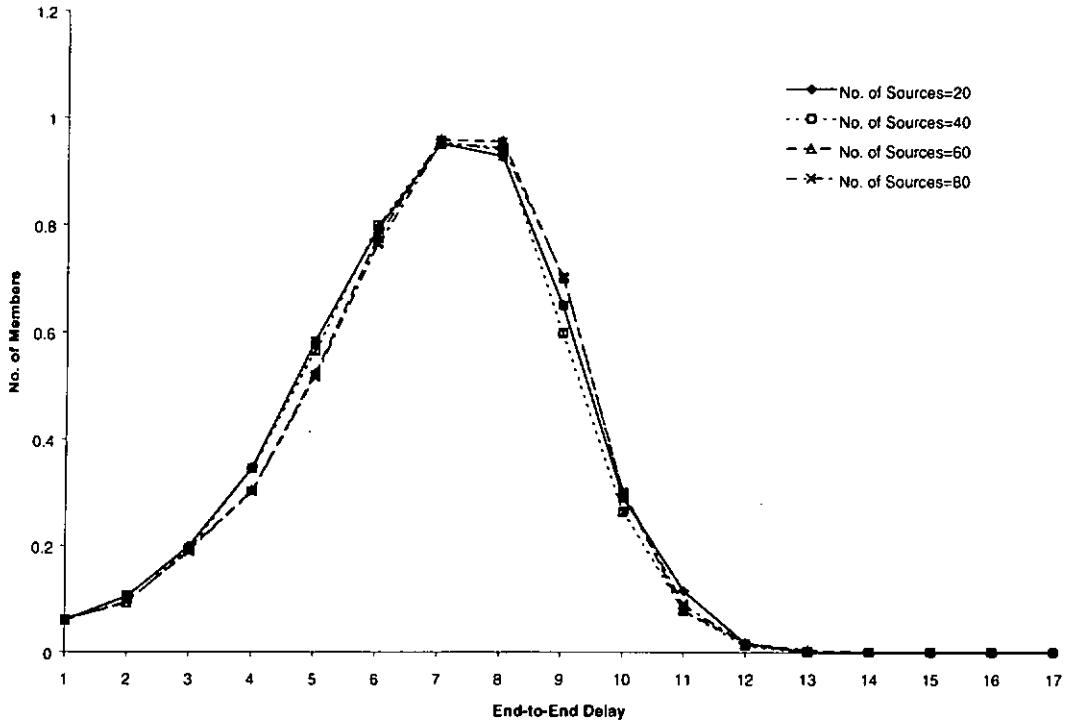
The end-to-end delay distribution in PIM-SM-t with group density of 0.05 is shown in Fig. 3.10 below.



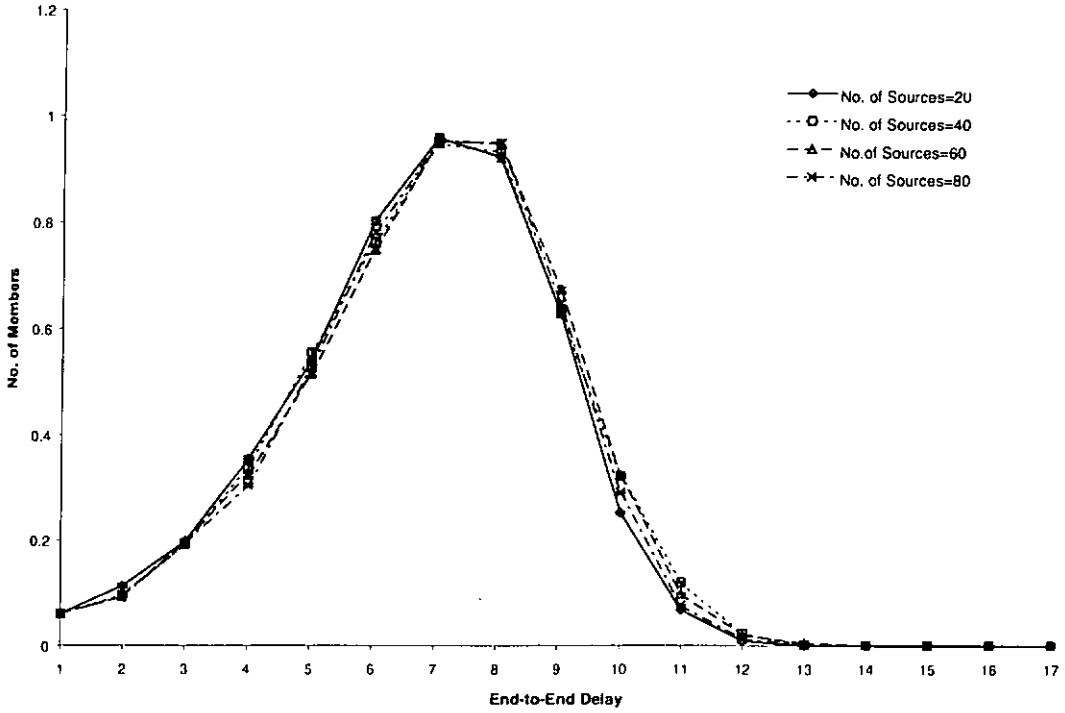
(a) Scenario 1



(b) Scenario 2



(c) Scenario 3

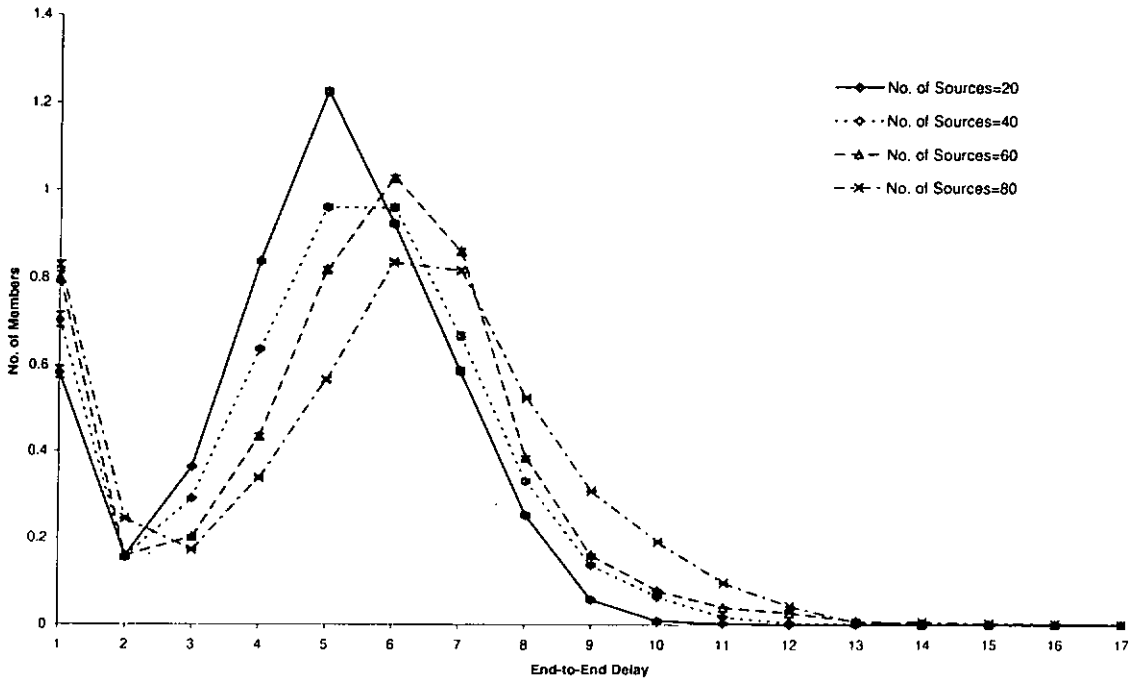


(d) Scenario 4

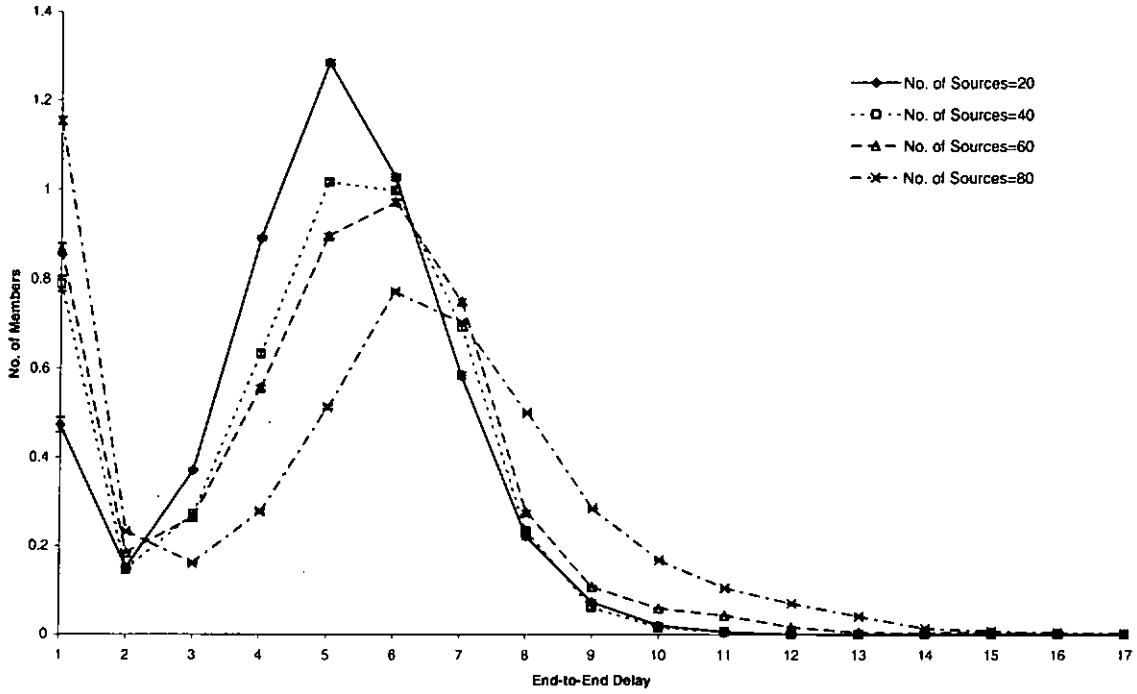
Fig. 3.10. End-to-end delay distribution of PIM-SM-t.

Similar to PIM-DM and PIM-SM-s, the numbers of group members in PIM-SM-t experiencing different end-to-end delay are more or less the same with different number of sources in different scenarios. This makes the average end-to-end delays with different number of sources in different scenarios to be more or less the same (Fig. 3.7). The end-to-end delay distributions are in the shape of a hill with the peak at 7-8 hops (Fig. 3.10). Most group members receive multicast packets from a source after 7-8 hops. It is less efficient than PIM-DM and PIM-SM-s in which most members receive packets after 3-4 hops and 6-7 hops respectively. There is about 0.06 member out of 5 group members experiencing 1 hop end-to-end delay from a source.

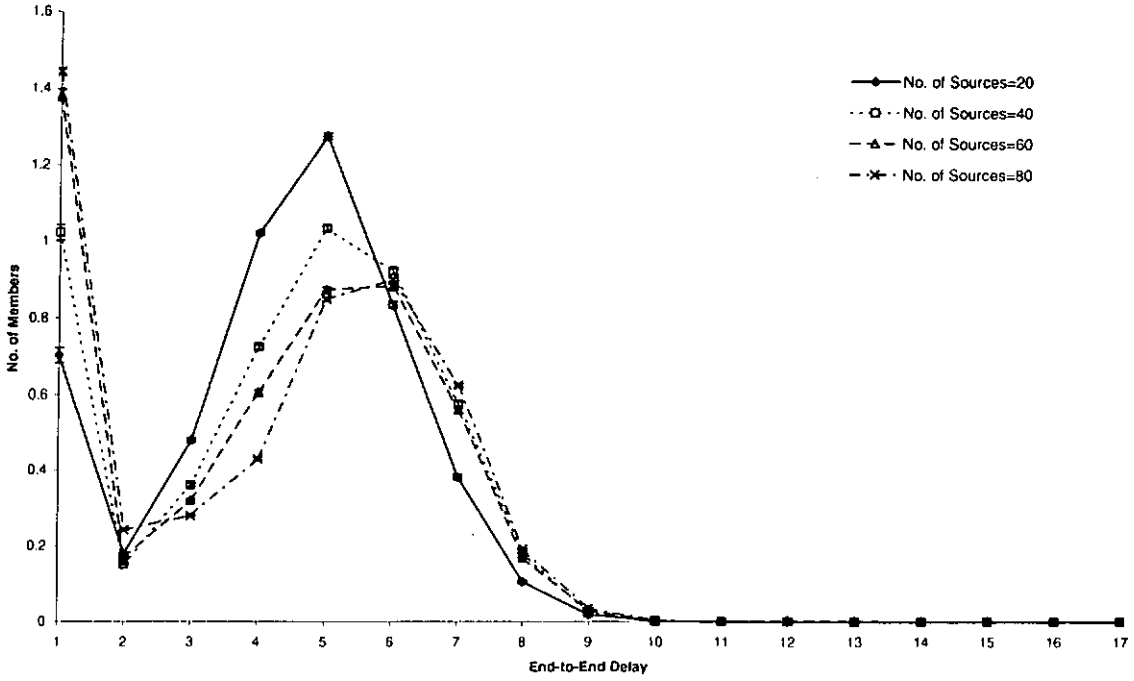
The end-to-end delay distribution in ASCORT with group density of 0.05 is shown in Fig. 3.11. Unlike PIM-DM, PIM-SM-s, and PIM-SM-t, the numbers of group members in ASCORT experiencing different end-to-end delay varies with different number of sources in different scenarios. The end-to-end delay distributions are in the shape of a hill with the peak at 5-6 hops (Fig. 3.11). As the number of sources increases, the peak shrinks and the number of group members experiencing 1 hop end-to-end delay from a source increases. Specifically, 0.58, 0.86, 1, and 1.2 members out of 5 members receive multicast packets with 1 hop end-to-end delay from a source when there are 20, 40, 60, and 80 sources in the multicast group. Such figures are significantly more than what exhibit in PIM-DM (0.23), PIM-SM-s (0.1), and PIM-SM-t (0.06).



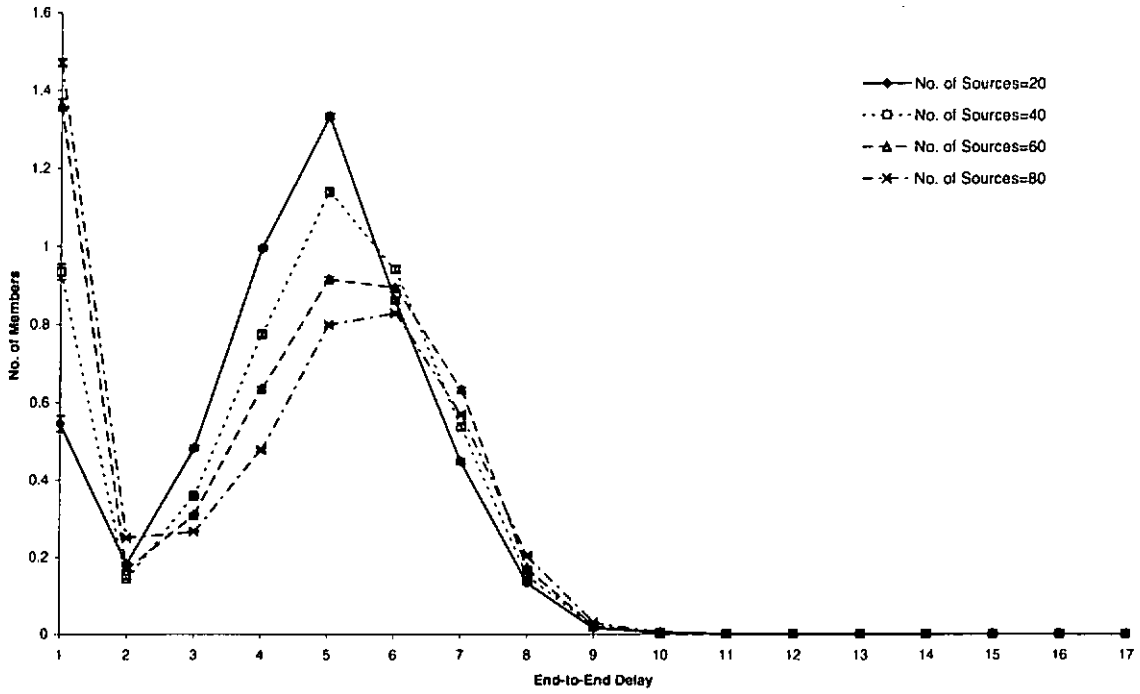
(a) Scenario 1



(b) Scenario 2



(c) Scenario 3



(d) Scenario 4

Fig. 3.11. End-to-end delay distribution of ASCORT.

3.5.4 Overhead of Control Messages

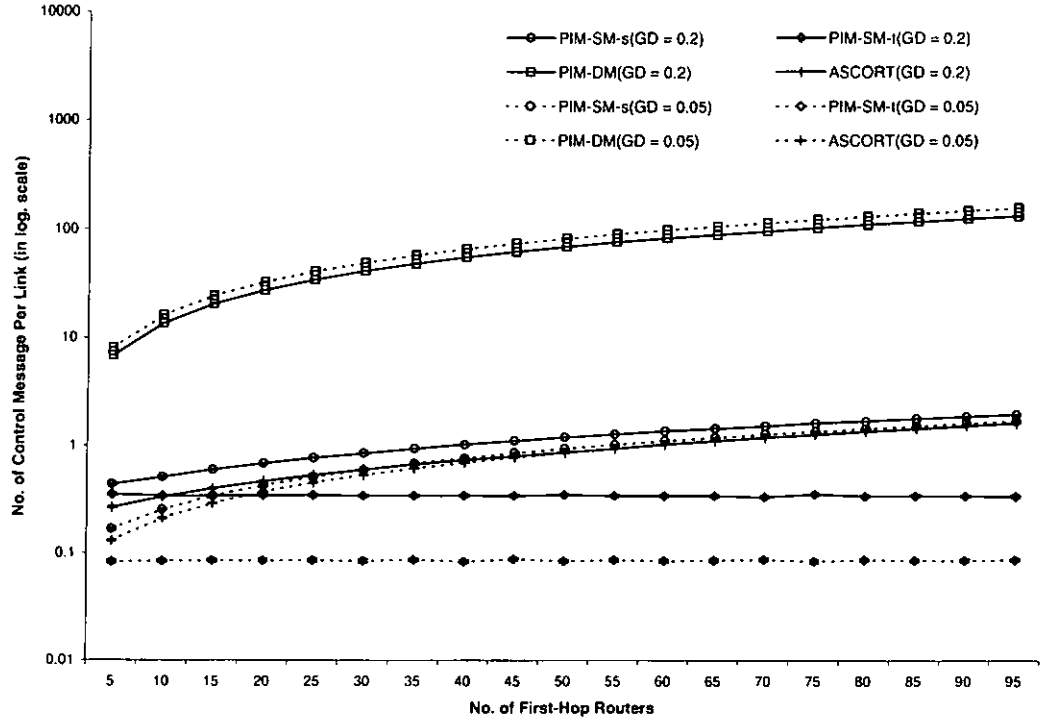
The control overhead of a multicast routing protocol is measured in terms of the number of control messages per link required to construct multicast distribution trees. This provides the overhead each link spends on constructing multicast distribution trees.

Given a random network, $N = (V, E)$, and a multicast group, g, multicast routing protocols based on shared trees construct one multicast distribution tree. On the other hand, multicast routing protocols based on source-specific trees build $|S_g|$ multicast distribution trees where S_g is the set of sources in the multicast group g. Let T be the set of multicast distribution trees built by a multicast routing protocol. The control overhead of the multicast routing protocol is defined as

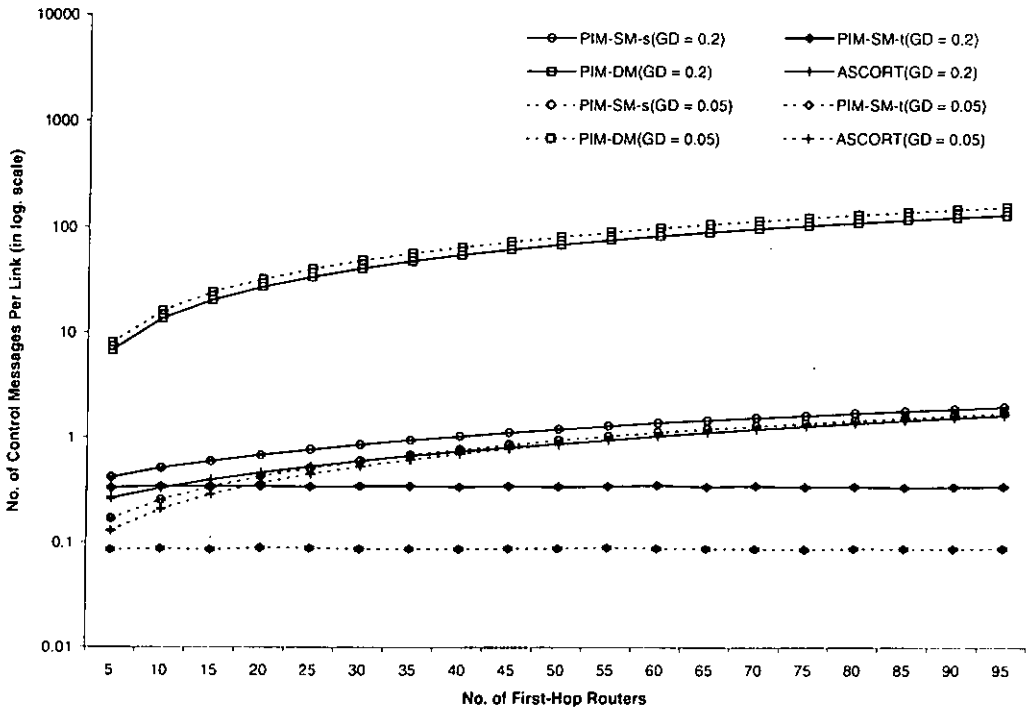
$$\text{control overhead} = \frac{\sum_{\forall t \in T} C(t)}{|E|} \tag{3.4}$$

where $C(t)$ is the number of control messages required to construct the multicast distribution tree t. In particular, PIM-DM uses *Prune* messages, PIM-SM-s and PIM-SM-t utilize *Join/Prune* messages, ASCORT employs *GRAFT*( ) and *PRUNE*( ) messages.

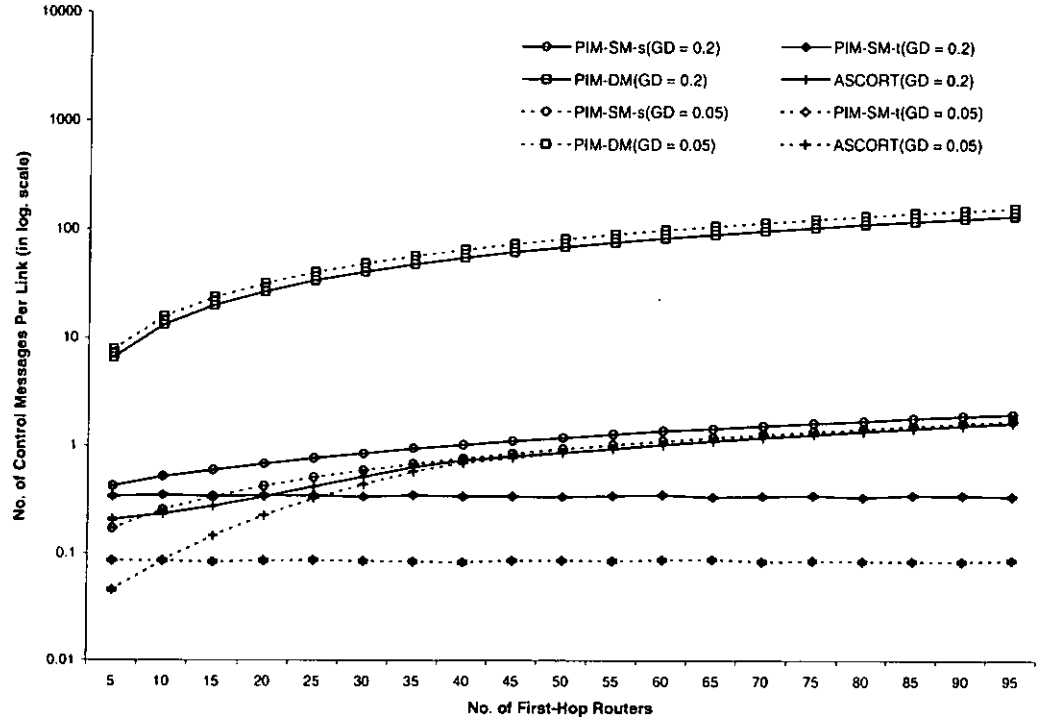
Fig. 4.12 shows the overhead of PIM-DM, PIM-SM-s, PIM-SM-t, and ASCORT in different distributions of first-hop routers and member routers in group density of 0.05 and 0.2.



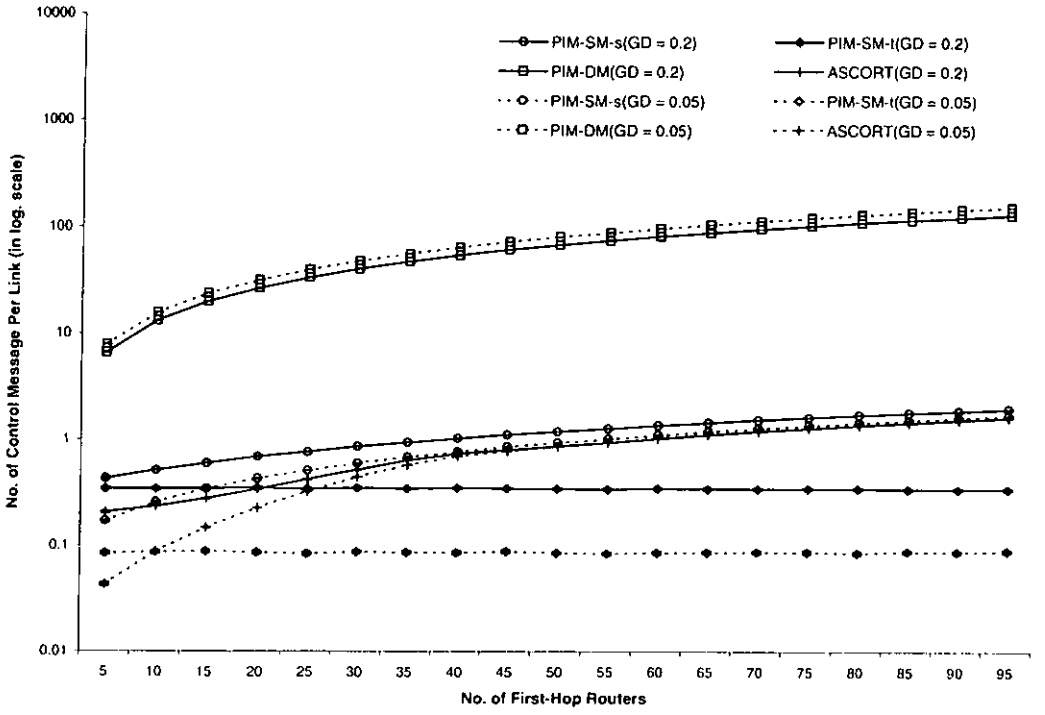
(a) Scenario 1



(b) Scenario 2



(c) Scenario 3



(d) Scenario 4

Fig. 3.12. Control overhead in different protocols.

As shown in Fig. 3.12, the performances of PIM-DM, PIM-SM-s, PIM-SM-t, and ASCORT in terms of overhead of control messages do not change significantly in the four different scenarios. Since the number of routers on the multicast distribution trees is more or less the same under different distribution of first-hop routers and member routers, the same amount of operations such as grafting and pruning is required to construct the multicast trees. As a consequence, the number of control messages is more or less the same in different scenarios.

The number of control messages used in PIM-DM is the largest among all the protocols because it requires each non-member leaf routers to send a *Prune* message upstream to prune the branches away. This resulted in a large amount of control messages flowing in the network. As the number of first-hop routers increases, the number of source specific trees increases and hence the number of non-member leaf routers increases. As a result, the *Prune* message generated by non-member leaf routers and hence the number of control messages used in PIM-DM increases as the number of first-hop routers increases. The number of control messages used under group density = 0.2 is less than that under group density = 0.05 because the number non-member leaf routers under group density = 0.2 is comparatively less than that under group density = 0.05.

Since a single shared tree is constructed in PIM-SM-t, the number of control messages used is more or less the same as the number of first-hop routers increases. In order to construct a shared tree, *Join/Prune* messages are sent from member routers to the RP. Therefore, the number of control messages under group density = 0.2 is larger than that under group density = 0.05. In addition to send *Join/Prune* messages from member routers to the RP to construct a shared tree, source specific paths are built by sending *Join/Prune* messages from the RP to first-hop routers in PIM-SM-s. It is for this reason that the number of control messages in PIM-SM-s is larger than that in PIM-SM-t. The number of control messages used in PIM-SM-s increases as the number of first-hop routers increases because a source specific path is built from the RP to each first-hop router.

In ASCORT, member routers send *GRAFT*( ) messages to their selected sources to construct reduced trees and first-hop routers send *GRAFT*( ) messages to the core to construct a bi-directional tree. The number of control messages used in ASCORT is less than that in PIM-SM-s because (i) member routers are connected to their nearest sources in terms of hop count in ASCORT and hence the number of hops between the member routers and its selected sources in ASCORT is significantly less than that between member routers and the RP in PIM-SM-s; and (ii) the core in ASCORT can adapt to the location of first-hop routers and hence the number of hops between the first-hop routers and the core in ASCORT is less than that between the RP and first-hop routers in PIM-SM-s. Let us note that reducing the number of hops between a member router and its selected source implies the number of links a *GRAFT*( ) message traversed is reduced. Similarly, reducing the number of hops between a first-hop router and the core implies the number of links a *GRAFT*( ) message traversed is reduced.

When compare ASCORT to PIM-SM-t, the number of control messages used in ASCORT is less than that in PIM-SM-t when the number of first-hop routers is small. Nevertheless, the number of control messages used in ASCORT is larger than that in PIM-SM-t as the number of first-hop routers increases. Although control messages are generated by first-hop routers to construct a bi-directional tree in ASCORT, the number of control messages used is comparatively small when compared to the number of control messages used to construct a shared tree in PIM-SM-t when the number of first-hop routers is small. Moreover, since the number of control messages used to construct reduced trees in ASCORT is significantly less than that used to construct a shared tree in PIM-SM-t, the total number of control messages used to construct a multicast distribution tree in ASCORT is less than that in PIM-SM-t when the number of first-hop routers is small. When the number of first-hop routers increases, the number of control messages used to construct the bi-directional tree increases. As a result, the number of control messages used to construct a multicast distribution tree in ASCORT is larger than that in PIM-SM-t as the number of first-hop routers increases.

In ASCORT, the number of control messages used under group density = 0.2 is larger than that under group density = 0.05. It is because the number of member routers under group density = 0.2 is larger than that under group density = 0.05 and hence the number of control messages generated by member routers to construct reduced trees under group density = 0.2 is larger than that under group density = 0.05. Let us note that the number of control message used in scenario 1 and 2 is larger than that in Scenario 3 and 4. It is because first-hop routers are distributed locally in Scenario 3 and 4 and hence the number of hops between first-hop routers and the core is smaller than that in Scenario 1 and 2. As a result, the number of control messages used to construct a bi-directional tree in Scenario 3 and 4 is less than that in Scenario 1 and 2.

3.5.5 Adaptability to Group Dynamics

To evaluate the performance of ASCORT under dynamic changes of group membership in the Internet applications over time, we run simulations in which sources and members join/leave a multicast group over time according to the Poisson distribution. The Poisson probability distribution is expressed as

$$f(x) = \frac{\lambda^x e^{-\lambda}}{x!} \tag{3.5}$$

In our simulations, it is possible that (i) a new source joins the multicast group; (ii) an existing source leaves the multicast group; (iii) a new member joins the multicast group; and (iv) an existing member leaves the multicast group. The above events are occurred according to the Poisson distribution with different value of λ. We denote the rate for a new source joins the

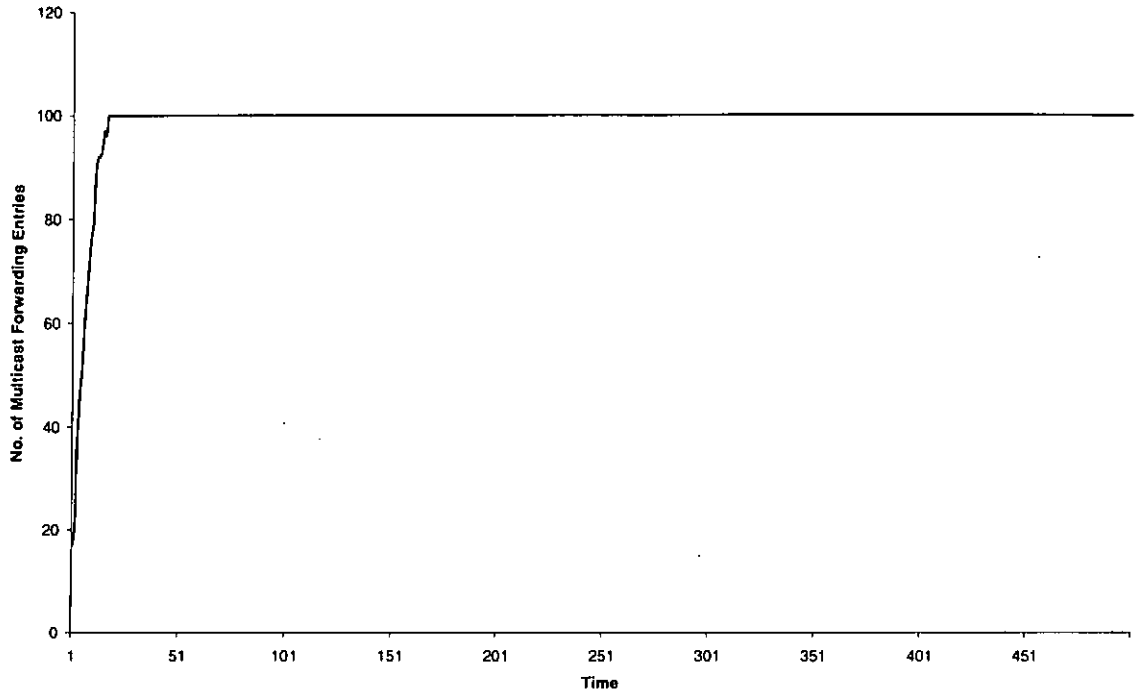
multicast group as λ_1, the rate for an existing source leaves the multicast group as λ_2, the rate for a new member to join the multicast group as λ_3, and the rate for an existing member leaves the multicast group as λ_4.

In the simulation experiments, we use four settings of these rates: (i) $\lambda_1 = 4$, $\lambda_2 = 1$, $\lambda_3 = 8$, $\lambda_4 = 2$; (ii) $\lambda_1 = 4$, $\lambda_2 = 1$, $\lambda_3 = 8$, $\lambda_4 = 4$; (iii) $\lambda_1 = 4$, $\lambda_2 = 2$, $\lambda_3 = 16$, $\lambda_4 = 4$; and (iv) $\lambda_1 = 4$, $\lambda_2 = 2$, $\lambda_3 = 16$, $\lambda_4 = 8$. It is important to note that the settings in case (i) and (ii) are the same except the rate for a member to leave in case (ii) is twice of that in case (i). Similarly, the settings in case (iii) and (iv) are the same except the rate for a member to leave in case (iv) is twice of that in case (iii). Let us further note that the rates for a source to join the multicast group in case (i)-(ii) and case (iii)-(iv) are the same whereas the rate for a source to leave the multicast group in case (iii)-(iv) is twice of that in case (i)-(ii).

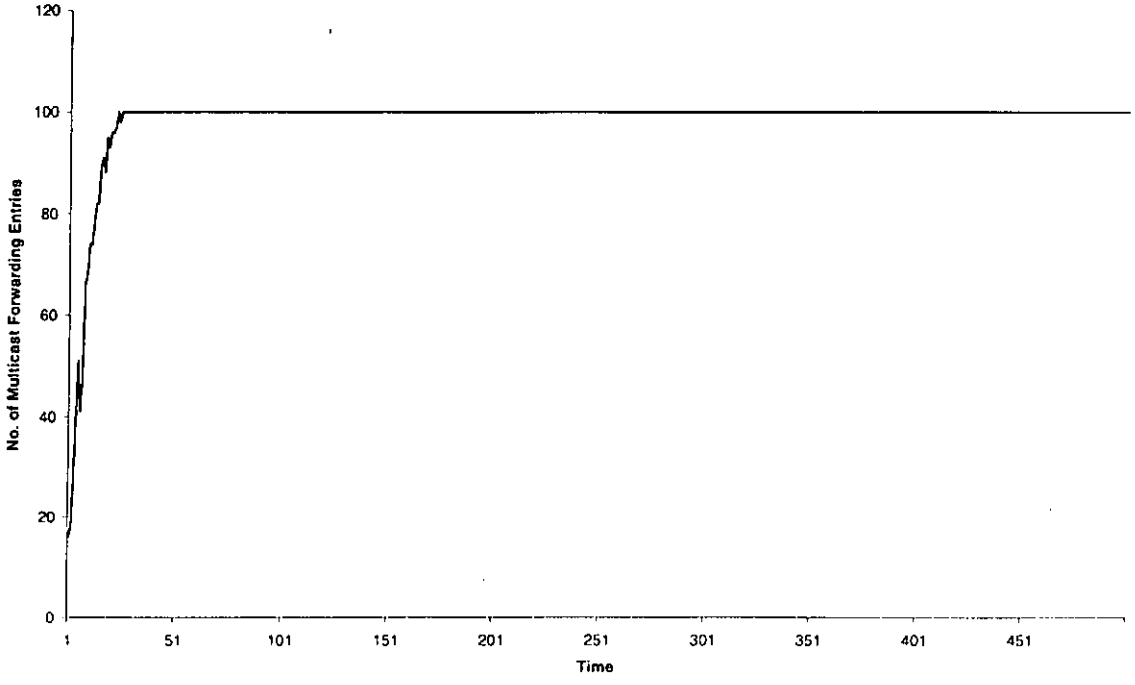
We measure the performance of ASCORT in terms of scalability, average end-to-end delay, and control overhead in every time unit. The scalability is the total number of multicast forwarding entries maintained in multicast routers lie on the multicast distribution tree constructed by ASCORT. The average end-to-end delay is the average number of hops from first-hop routers to member routers. The control overhead is the number of control messages per link generated by ASCORT to construct and maintain a multicast distribution tree. We run the simulations over 500 time units. Each member router maintains a timer to perform periodic refresh every 60 time units.

Fig. 3.13 shows the scalability of ASCORT in terms of number of forwarding entries maintained at multicast routers over time. As shown in Fig. 3.13, the number of multicast forwarding entries increases and then stabilizes at the value of 100. It is because the rate of source/member joins the multicast group is greater than that of source/member leaves. As time goes by, almost all the routers in the network become first-hop routers and/or member routers. This makes almost all the routers become on-tree routers. Since each on-tree router maintains a multicast forwarding entry and there are 100 routers in the network, there are 100 multicast forwarding entries in total.

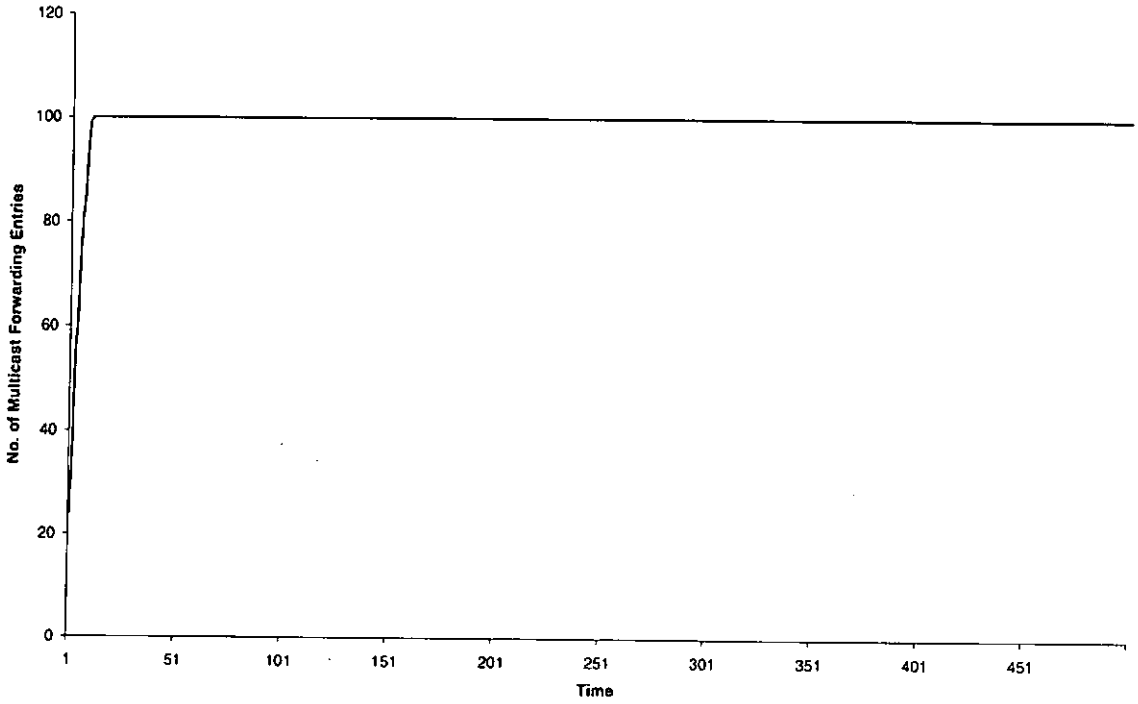
The number for multicast forwarding entries reaches 100 in Fig. 3.13(a) at time unit smaller than that in Fig. 3.13(b) because the rate of member leaves in Fig. 3.13(a) is faster than that in Fig. 3.13(b). Similarly, the number of multicast forwarding entries reaches 100 in Fig. 3.13(c) faster than in Fig. 3.13(d).



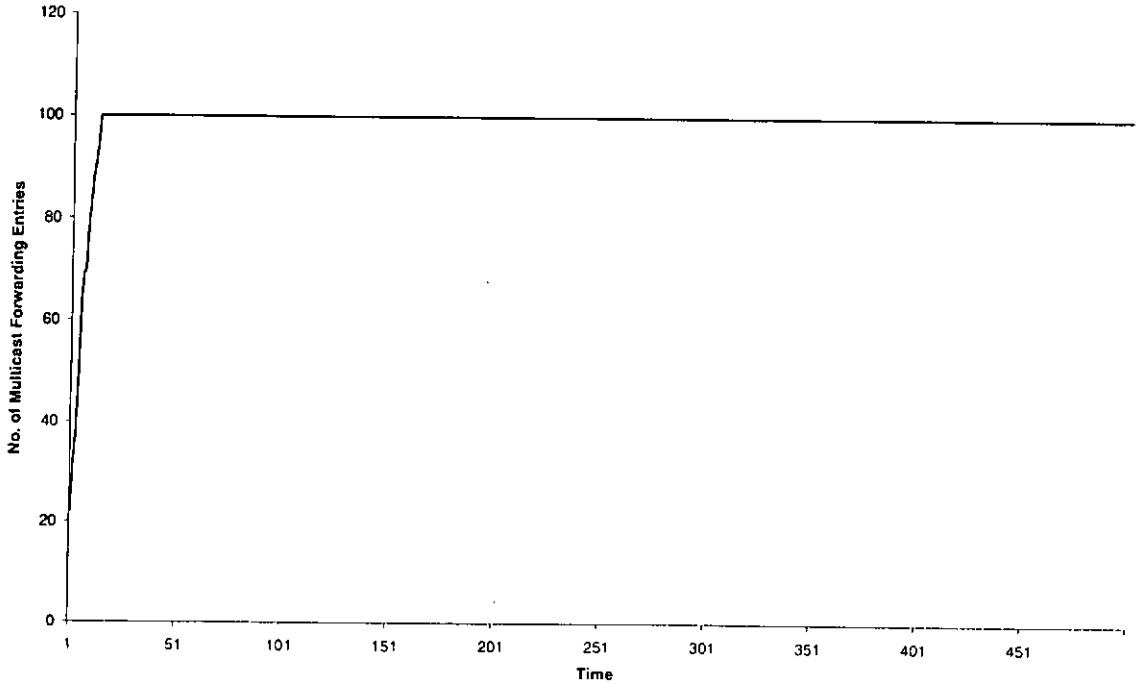
(a) $\lambda_1 = 4$, $\lambda_2 = 1$, $\lambda_3 = 8$, $\lambda_4 = 2$



(b) $\lambda_1 = 4$, $\lambda_2 = 1$, $\lambda_3 = 8$, $\lambda_4 = 4$



(c) $\lambda_1 = 4$, $\lambda_2 = 2$, $\lambda_3 = 16$, $\lambda_4 = 4$



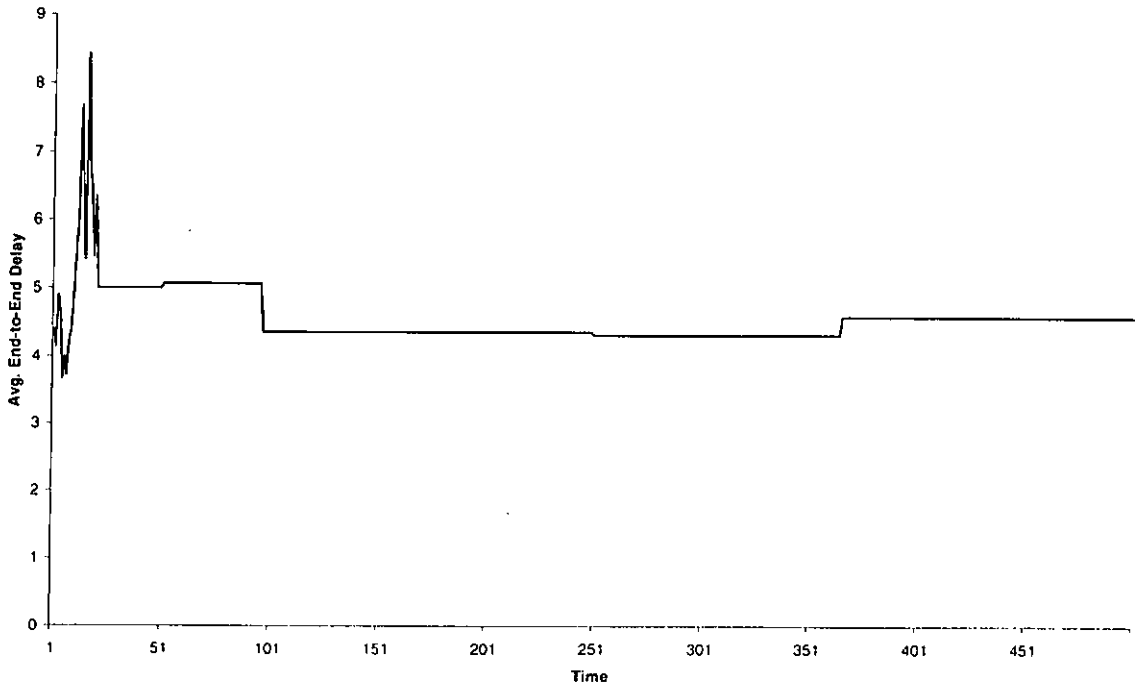
(d) $\lambda_1 = 4$; $\lambda_2 = 2$, $\lambda_3 = 16$, $\lambda_4 = 8$

Fig. 3.13. Scalability of ASCORT over time.

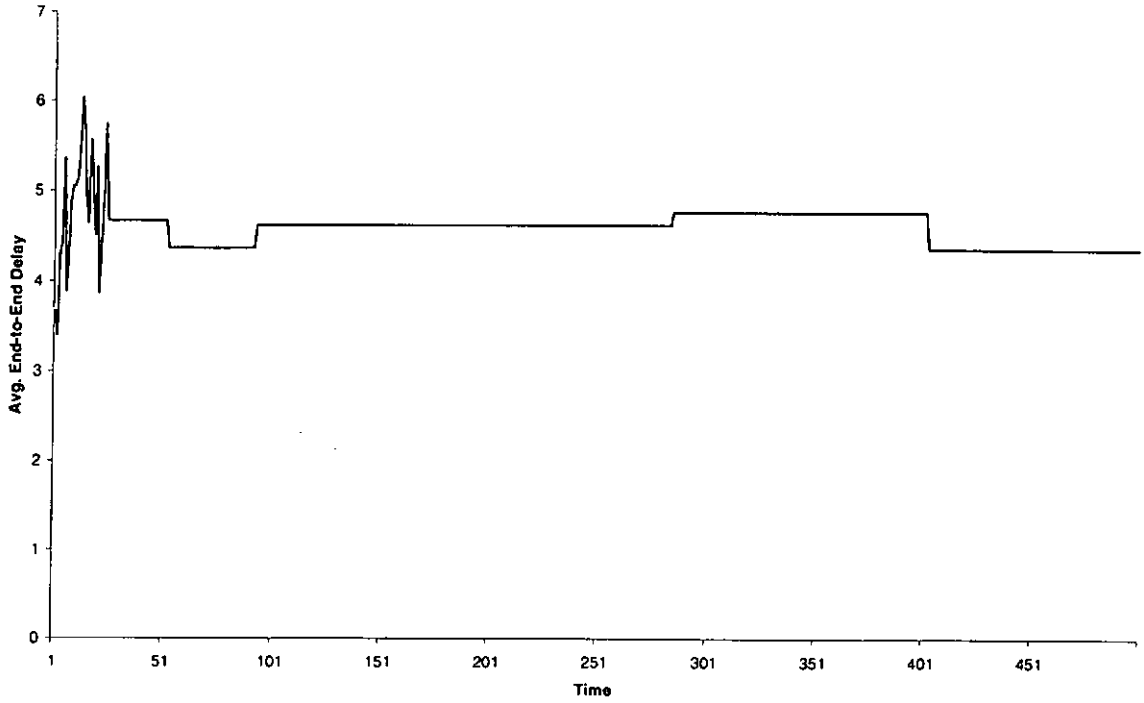
The average end-to-end delay of ASCORT over time is given in Fig. 3.14. It is important to note that the average end-to-end delay changes significantly when the base source leaves the multicast group. When the time unit is small (≤ 25), the average end-to-end delay fluctuated frequently because the base source has higher probability to leave the multicast group. As time goes by, the number of sources increases and the probability that the source which leaves the multicast group is the base source decreases. It is for this reason that the average end-to-end delay varies slowly when time unit is greater than 25.

The readers should note that the average end-to-end delay increases/decreases sharply and then becomes more or less the same when time unit is greater than 25. Such phenomena repeat in the course of the simulations. The sudden changes in average end-to-end delay are due to the leaves of the base source from the multicast group. Afterwards, the joins/leaves of sources/members have little effect on the average end-to-end delay because almost all routers are already on-tree no matter how the group memberships are changed.

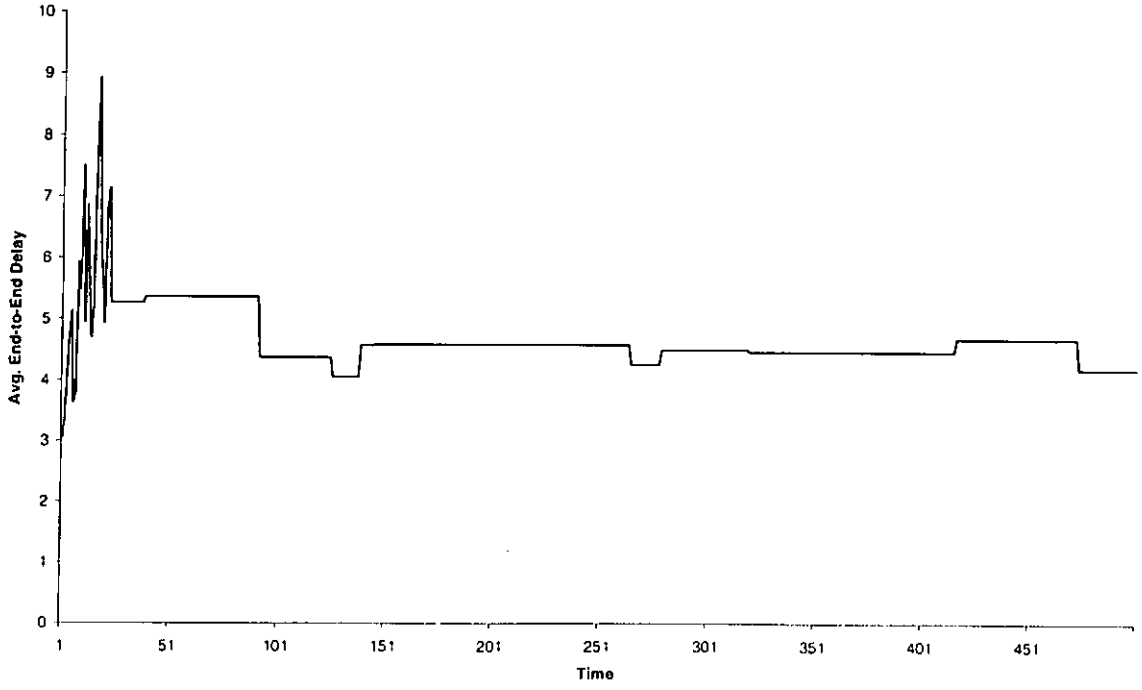
Let us consider Fig. 3.14(a) and Fig. 3.14(c). The average end-to-end delay changes more frequently in Fig. 3.14(c) than that in Fig. 3.14(a) because the rate of source leaves in Fig. 3.14(c) is faster than that in Fig. 3.14(a). Consequently, the probability that the source, which leaves the multicast group, is the base source is higher in Fig. 3.14(c) than that in Fig. 3.14(a). Similar observations can be found in Fig. 3.14(b) and Fig. 3.14(d).



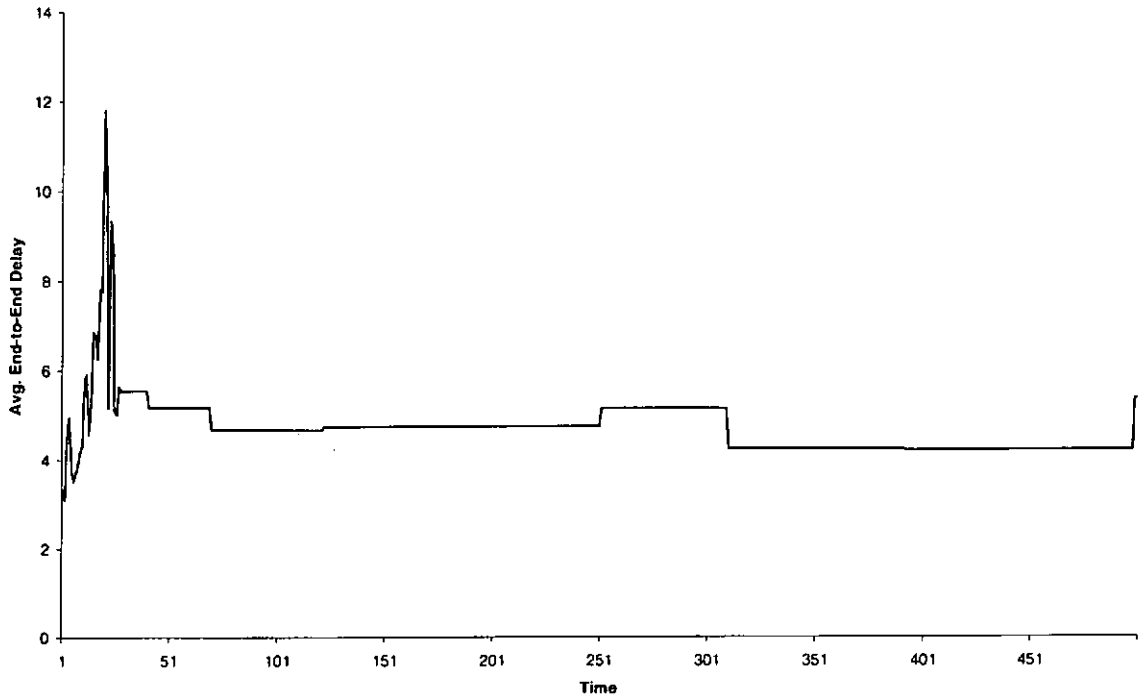
(a) $\lambda_1 = 4$, $\lambda_2 = 1$, $\lambda_3 = 8$, $\lambda_4 = 2$



(b) $\lambda_1 = 4$, $\lambda_2 = 1$, $\lambda_3 = 8$, $\lambda_4 = 4$



(c) $\lambda_1 = 4$, $\lambda_2 = 2$, $\lambda_3 = 16$, $\lambda_4 = 4$

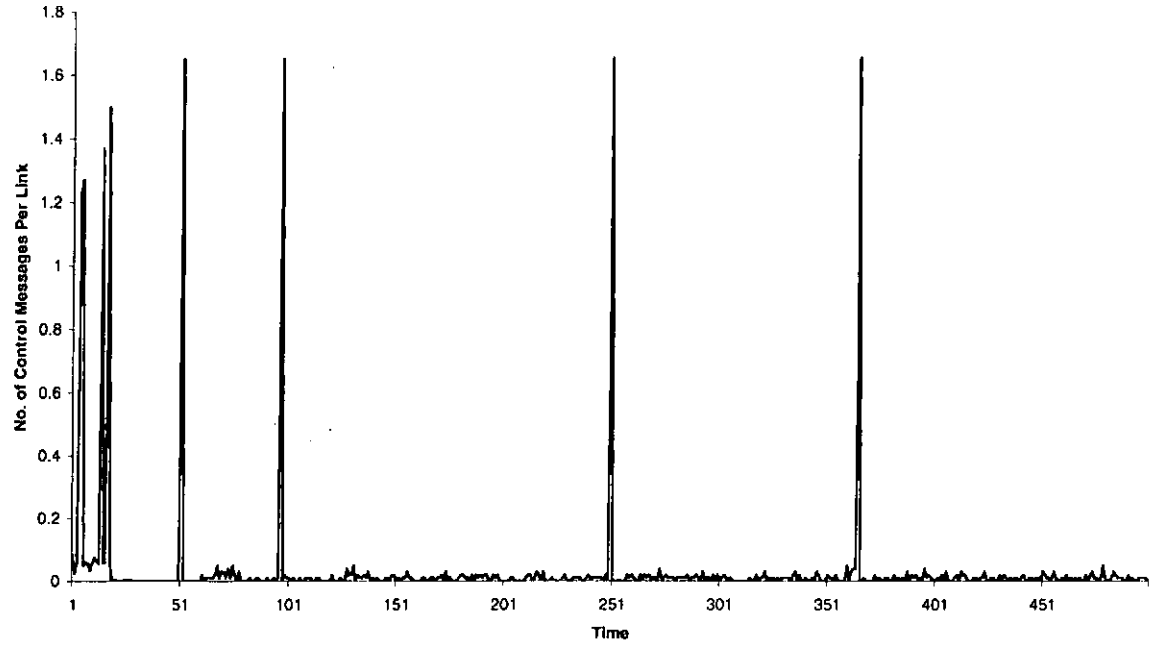


(d) $\lambda_1 = 4$, $\lambda_2 = 2$, $\lambda_3 = 16$, $\lambda_4 = 8$

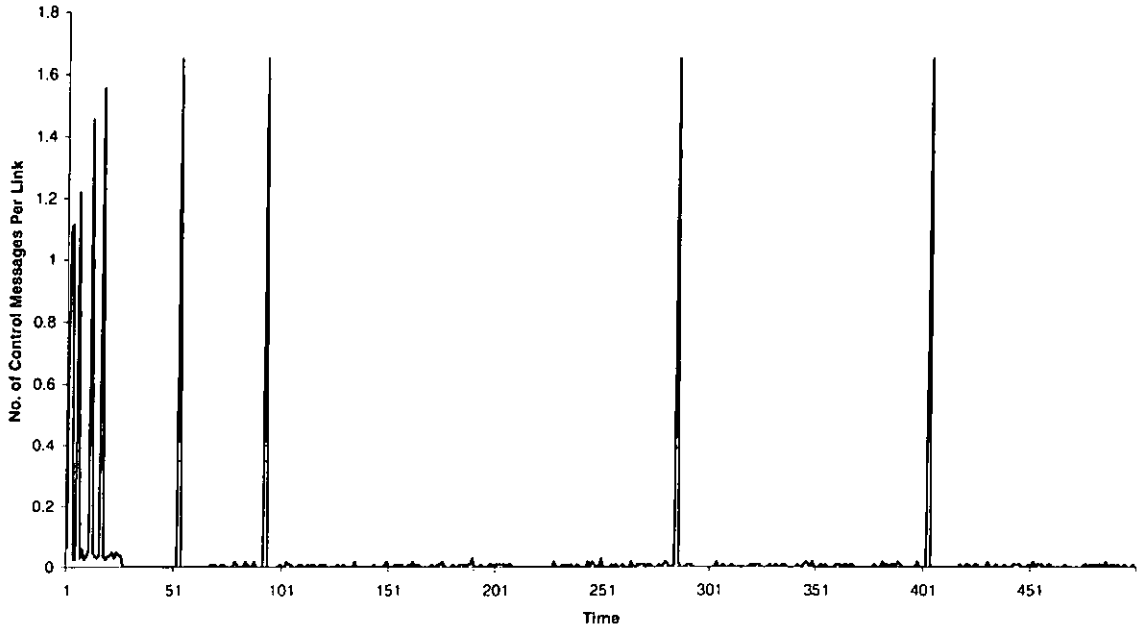
Fig. 3.14. Average end-to-end delay of ASCORT over time.

The control overhead of ASCORT over time is given in Fig. 3.15. Similar to the average end-to-end delay, the control overhead changes significantly when the base source leaves the multicast group. When the time unit is small (≤ 25), the control overhead fluctuated frequently because the base source has higher probability to leave the multicast group. As time goes by, the number of sources increases and the probability that the source, which leaves the multicast group, is the base source decreases. Consequently, the control overhead varies slowly when time unit is greater than 25.

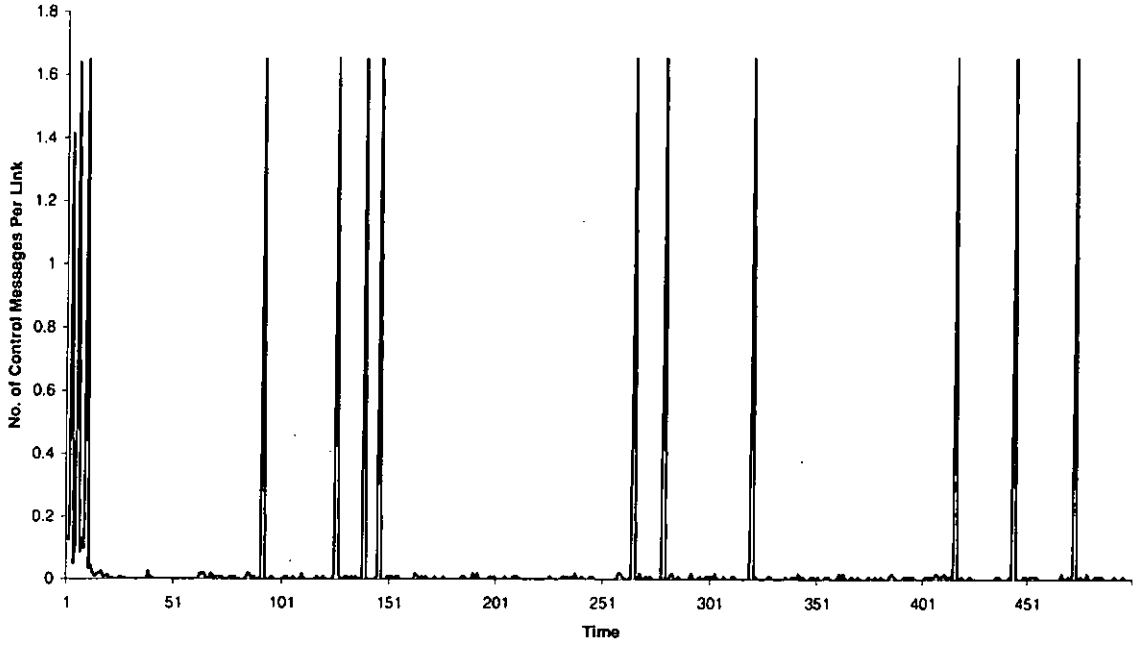
Let us consider Fig. 3.15 (a) and Fig. 3.15 (c). The control overhead changes more frequently in Fig. 3.15 (c) than that in Fig. 3.15 (a) because the rate of source leaves in Fig. 3.15 (c) is faster than that in Fig. 3.15 (a) and hence the probability that the base source leaves is higher in Fig. 3.15 (c) than that in Fig. 3.15 (a). The effects of the leaves of the base source on the control overhead are represented by the "spikes" in Fig. 3.15. Similar observations can also be found in Fig. 3.15 (b) and Fig. 3.15 (d).



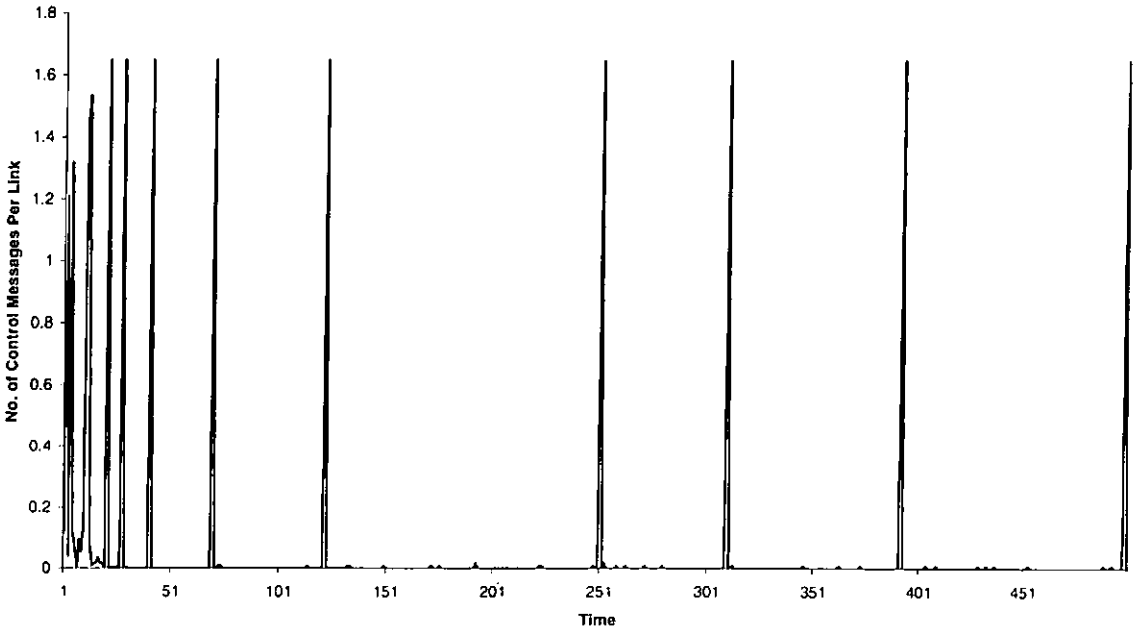
(a) $\lambda_1 = 4$, $\lambda_2 = 1$, $\lambda_3 = 8$, $\lambda_4 = 2$



(b) $\lambda_1 = 4$, $\lambda_2 = 1$, $\lambda_3 = 8$, $\lambda_4 = 4$



(c) $\lambda_1 = 4$, $\lambda_2 = 2$, $\lambda_3 = 16$, $\lambda_4 = 4$



(d) $\lambda_1 = 4$, $\lambda_2 = 2$, $\lambda_3 = 16$, $\lambda_4 = 8$

Fig. 3.15. Control overhead of ASCORT over time.

3.6 Summary

We have introduced a novel protocol called ASCORT to construct multicast distribution trees in this chapter. Instead of building source-specific trees and shared trees, the multicast distribution trees are composed of a forest of reduced trees and a bi-directional tree. The technique used in constructing reduced trees is able to partition group members into different disjoint subsets in an effective manner. It is this technique allowing each multicast router to maintain at most one routing entry. The simulation results showed that ASCORT is able to achieve better scalability and end-to-end delay when compared to PIM-SM. Furthermore, the results also illustrated that ASCORT is very efficient in the construction of multicast distribution trees in terms of number of control messages. The simulation experiments demonstrated that ASCORT could be used as an alternative way to enable multicast routing for large-scale real-world applications. We summarize the performance of PIM-DM, PIM-SM-s, PIM-SM-t and ASCORT in Table 3.2.

	PIM-DM	PIM-SM-s	PIM-SM-t	ASCORT
Scalability	Poor	Fair	Excellent	Good
Efficient Packet Distribution	Excellent	Fair	Poor	Good
Effective Method for Core/ RP Selection	Not Applicable	Poor	Poor	Good
Minimal Control Overhead	Poor	Fair	Excellent	Good
Adaptability of Group Dynamics	Poor	Good	Good	Good
Independence of Unicast Routing Protocols	No	Yes	Yes	Yes
Support for Quality of Service	Yes	Yes	No	Yes

Table 3.2. Comparison of PIM-DM, PIM-SM and ASCORT.

Chapter 4

A Tree Switching Protocol for Multicast State Reduction

In this chapter, we propose a new protocol called Tree Switching Protocol (TSP) to reduce multicast forwarding states required for a forest of shared trees or a forest of source-specific trees. The protocol accomplishes this goal by selecting a base multicast tree and "switching" other multicast trees to the base tree. If the multicast trees are not overlapped with the base tree, the tree switching operation will result in more overlapping among the multicast trees and a net reduction in multicast state. A further reduction in the state is made possible by applying state aggregation (e.g. [Thaler and Handley 2000; Radoslavov, Estrin, and Govindan 1999]) to the overlapped tree branches. The simulation results for PIM-SM show that the tree switching operation alone could result in a very significant state reduction. With additional mechanisms (e.g. leaky and non-leaky state aggregation) to reduce the states for overlapped tree branches, the state requirement can be further reduced. The TSP is a distributed protocol running on top of any protocol independent multicast routing protocols. It is also loop-free and very efficient.

4.1 Introduction

Scalability is one of the important criterions for evaluating a particular multicast routing protocol. Scalability is used to measure the amount of state information required to maintain in multicast routers to support multicasting. The smaller amount of state information requires, the better scalability of a multicast routing protocol is. Since the source-specific approach requires a separate tree for each active source, this approach is not scalable. Although, the shared tree approach is more scalable than the source-specific approach since only one tree is required for any number of sources, large amount of state information is still required when the number of multicast groups is large. The lack of scaling ability to scale to both number of sources and number of multicast groups could seriously inhibit the deployment of multicast service in the Internet because there is no "nature" limit to the number of concurrent multicast groups.

In this chapter, we propose a novel approach to reduce the multicast state requirement. Our approach is based on the following observation. The number of multicast trees (whether shared trees or source-specific trees) existing concurrently within a domain or in the Internet backbone could be very large as multicast-based applications become more popular.

Consequently, it is likely that two or more multicast trees could partially overlap with each other. In other words, a multicast router may belong to several multicast trees. Based on this observation, we propose a tree switching operation to reduce the total multicast state requirement for a forest of multicast trees. The idea is that a member router could select a multicast tree out of all the trees that it belongs as a base multicast tree. And it will attempt to switch other trees to the base tree. If successful, all other tree branches, except for the base tree, can be pruned away. If the number of forwarding entries pruned away is higher than the number of new entries added to the base tree, the tree switching will result in a net reduction in the amount of states. As we shall see from the simulation results later, the tree switching operation could indeed reduce the multicast state significantly. Moreover, the tree switching also increases the overlapping among multicast trees. A further state reduction can be accomplished by applying leaky or non-leaky forwarding state aggregation to the overlapped tree branches. We will define the meaning of tree overlapping more precisely in the next section.

The main contribution is to propose a new protocol called Tree Switching Protocol (TSP) for performing the tree switching operation. For the overlapped tree branches, the protocol's role is to trigger state aggregation, if available, for the overlapped branches. This approach is most suitable for both intra-domain and inter-domain levels because those multicast sessions tend to be long-lived and scalability is a prime concern.

To be more effective, our protocol is able to operate in a distributed manner. Since there is no single point of failure and the communication cost for a distributed protocol is usually lower than that of a centralized protocol, it is more desirable to have a distributed protocol for the Internet applications.

Moreover, our protocol is free of any routing loop. Since the Internet applications of multicast nature tend to be real-time and bandwidth intensive, loops in multicast routing duplicate looping packets for a large volume of packets. The multitude of duplicate packets results in the consumption of network bandwidth and the increase in queuing delay along a subtree of the multicast distribution tree. To provide multicast service effectively, the multicast distribution trees should be free of any loop when no loop is formed in unicast routing.

Furthermore, our protocol is able to run on top of any protocol independent multicast routing protocol. Our protocol only make use of the entries maintained in multicast forwarding tables but not dependent on a particular protocol to compute the tables.

The rest of this chapter is organized as follows. In Section 4.2, we provide an overview of the TSP and we will also discuss the state aggregation for overlapped tree branches. In Section 4.3, we describe the protocol in details and we will use PIM-SM as the underlying multicast routing protocol. In Section 4.4, we will show that the TSP is loop-free and incurs no packet loss. In Section 4.5, we will study the effect of TSP on the degree of tree overlapping, multicast state

reduction, end-to-end delay, and control overhead by simulation experiments. Finally, we summarize this chapter in Section 4.6.

4.2 An Overview of Tree Switching Protocol

To give an overview on how TSP operates, we use PIM-SM as the underlying multicast routing protocol. That is, we assume that there are already a number of shared trees, each of them has a *rendezvous point* (RP) as the tree root and is identified by a unique class D address. We employ the following notations for the rest of this chapter.

1. T, G: The set of shared trees and the corresponding set of multicast addresses under consideration.
2. T_i, g_i: The i-th shared tree with multicast address g_i.
3. (*, g_i, I_i, O_i): A multicast routing entry for multicast group g_i. The first parameter indicates the source host's address. This parameter is a wildcard for shared trees. I_i is an incoming interface, and O_i is an outgoing interface list for the multicast address g_i.

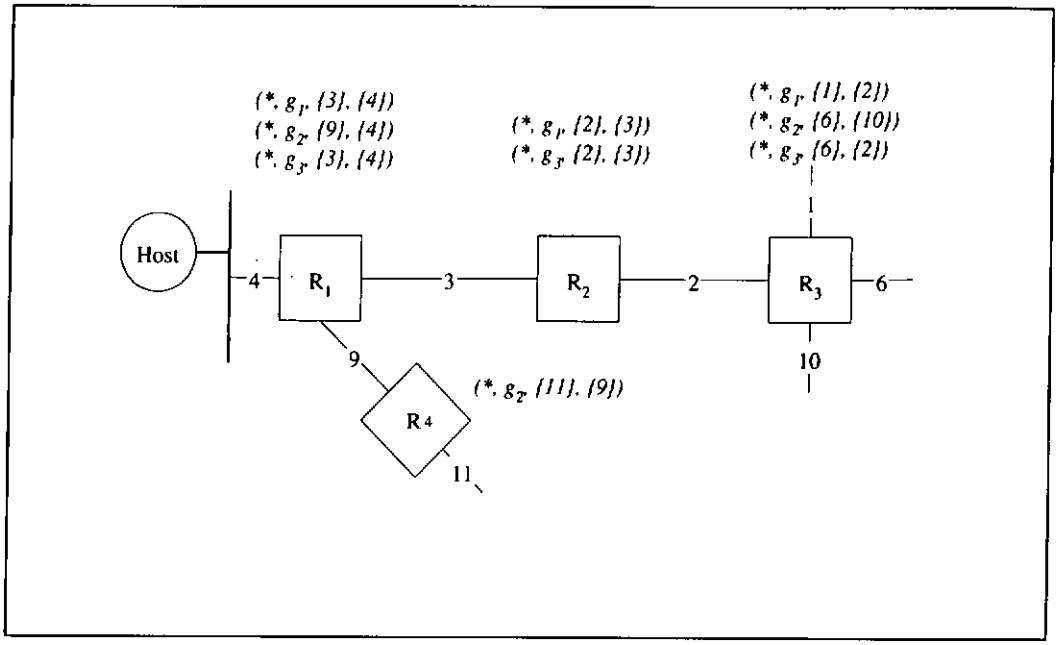
4.2.1 An Example

The key ideas behind TSP can be best explained by the example in Fig. 4.1 below. There are three multicast groups g_1, g_2, and g_3 and we show parts of T_1, T_2, and T_3 in the figure. In Fig. 4.1(a), R_1 is a member router for all three groups whereas the other three routers are on-tree routers. Moreover, R_1 is a leaf router on T_1, T_2, and T_3. We first define the meaning of tree overlapping. Multicast trees T_j and T_k are said to be *overlapped* on router R_i if and only if $I_j = I_k$ and $O_j = O_k$ in the two corresponding routing entries in R_i; otherwise they are not overlapped on R_i. Therefore in this example only T_1 overlaps with T_3 on R_1 and R_2.

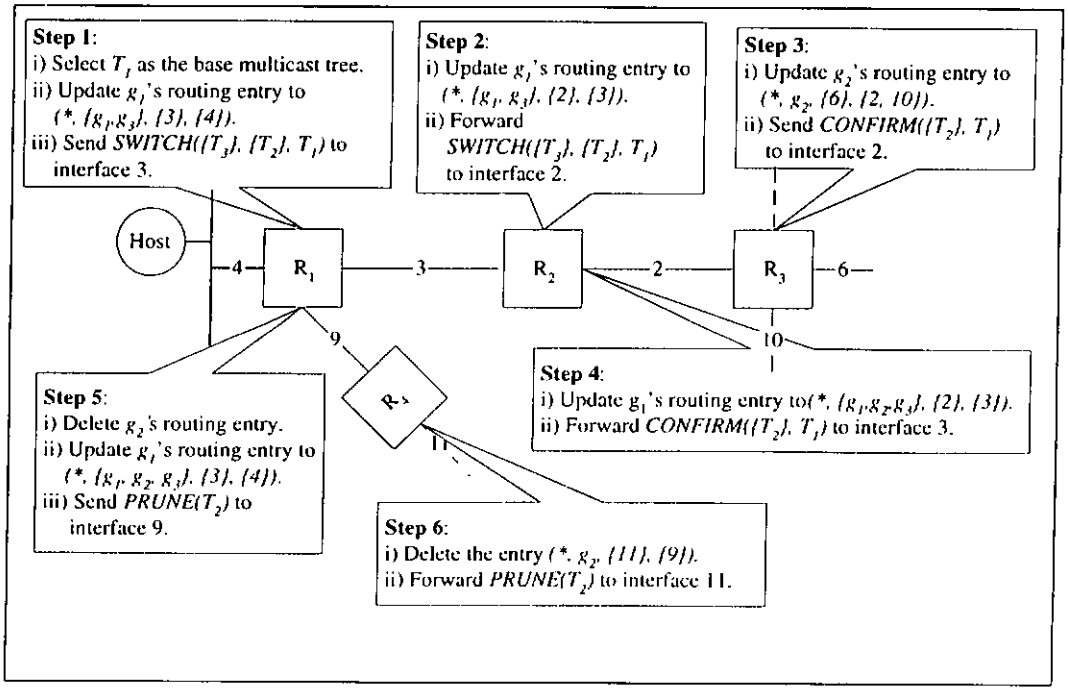
In this particular case, R_1 will attempt to switch $T_{2,3}$ to T_1. We call T_1 a *base multicast tree* and the other two *target multicast trees*. The whole process here involves three protocol messages: *SWITCH*( ), *CONFIRM*( ), and *PRUNE*( ). We will delay the detailed protocol description to the next section. For the time being, it suffices to understand that the *SWITCH*( ) message is a request message for switching T_2 and T_3 to T_1. Since T_1 overlaps with T_3 on R_1, tree switching is not necessary for T_3. Instead, in Fig. 4.1(b), we show that the protocol triggers R_1 to combine both entries into a logical entry. All of the above take place as the step 1 in the figure.

The *SWITCH*( ) message also triggers R_2 to combine the entries for g_1 and g_3 since T_1 also overlaps with T_3 on R_2 (step 2). When the message arrives at R_3, R_3 responds with a *CONFIRM*( ) message back to R_1 and the message confirms that T_2 can be switched to T_1 since R_3 is also on T_2. At the same time, R_3 adds the interface 2 to the g_2's entry so that downstream routers could receive multicast packets for g_2 (step 3). By receiving the *CONFIRM*( ) message, the interior router R_2

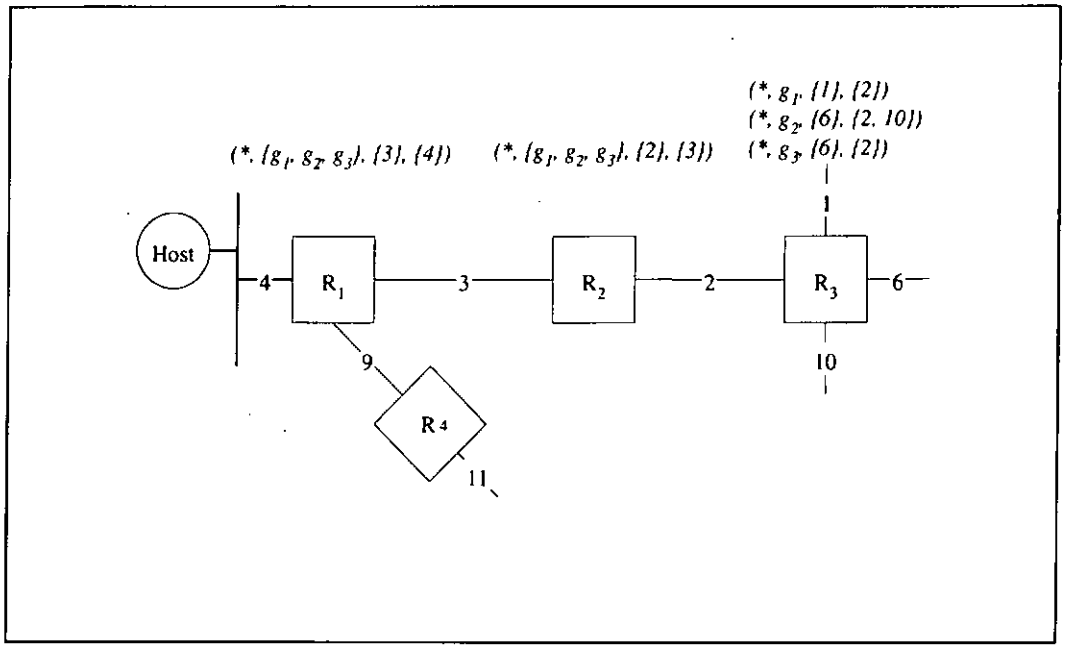
includes g_2 into its multicast forwarding entry since all three trees overlap with each other on this router (step 4).



(a) The original multicast routing states.



(b) Operation of the Tree Switching Protocol.



(c) Multicast routing states after tree switching.

Fig. 4.1. An example to illustrate tree switching operation.

When the message eventually arrives at R_1, it updates g_2's entry accordingly and sends a *PRUNE*( ) message to interface 9 to prune away T_2's branch as far as possible (step 5). R_4 is required to prune away g_2's entry upon receiving the *PRUNE*( ) message and possibly forwards the message to the upstream router (step 6). The tree switching operation is thus successful and the resulted forwarding entries are shown in Fig. 4.1(c).

4.2.2 Tree Overlapping and State Reduction

In Fig. 4.1(c), we depict that entries for overlapped trees as a single, "logical" entry since they share the same incoming and outgoing interfaces. But the actual storage requirement for such a logical entry is generally more than that for a single forwarding entry for a multicast group. One way of reducing the storage requirement for the logical entry is to "aggregate" redundant information like the incoming and outgoing interfaces. Nevertheless, there are still other state information about multicast groups, such as the soft-state timer value, need to be retained for each individual group. Moreover, the actual storage reduction in the multicast state also depends on the internal implementation details. Thus it is not straightforward to compute the actual amount of state reduction using the state aggregation approach. Instead, we will present simulation results in Section 4.5 on the multicast state reduction under different state aggregation ratios for overlapped trees.

4.3 Tree Switching Protocol in Details

The TSP is a distributed protocol. Therefore several tree switching operations may be active at any time. In describing the protocol operation, we will focus on a designated router R_i that initiates the protocol. The TSP distinguishes whether a designated router is a member router or an on-tree router. Only member router could initiate the TSP. In Fig. 4.2, we show the state diagrams for an interior router that receives a tree switching request from downstream (Fig. 4.2(b)), and for member routers (Fig. 4.2(a)). There are three router states, namely *Idle*, *Wait* and *Confirm*, for the former, and there is an additional *Wake-up* state for the latter. The state diagram here is also associated with a particular tree switching request. Thus, a particular designated router may follow the state diagram in Fig. 4.2(a) for a set of multicast trees but follows Fig. 4.2(b) for another set of multicast trees. Moreover, the state transitions are triggered either by timers or by the *SWITCH*( ), *CONFIRM*( ), and *PRUNE*( ) messages.

4.3.1 Wake-up Procedure

Every member router is equipped with a wake-up timer and the timer expires after a random period of time. The timer expiration "wakes" up a member router by making a state transition to the *Wake-up* state. We let $\boldsymbol{T}(i) \subseteq \boldsymbol{T}$ be the set of multicast trees that R_i is part of. The member router will perform the following when it enters into the *Wake-up* state.

1. R_i first discovers the set of multicast trees that it is part of and for which it is a leaf router. We label this set by $\boldsymbol{T}_l(i) \subseteq \boldsymbol{T}(i)$. The wake-up is considered successful if R_i is on at least two trees and $\boldsymbol{T}_l(i)$ is not empty; otherwise it goes back to the *Idle* state.

2. R_i selects a base multicast tree from $\boldsymbol{T}(i)$ for tree switching and we denote this tree as $T_b(i) \in \boldsymbol{T}(i)$. We do not require R_i to be a leaf router on $T_b(i)$. However, in this chapter we assume $T_b(i) \notin \boldsymbol{T}_l(i)$ except when all the trees in $\boldsymbol{T}(i)$ are also members of $\boldsymbol{T}_l(i)$, that is, $\boldsymbol{T}(i) = \boldsymbol{T}_l(i)$. By selecting the base tree in this way, we could switch more target trees to a base tree. If indeed $T_b(i) \in \boldsymbol{T}_l(i)$, we need to remove $T_b(i)$ from $\boldsymbol{T}_l(i)$ for the remaining tasks.

3. Based on $T_b(i)$ and $\boldsymbol{T}_l(i)$, R_i determines the set of trees in $\boldsymbol{T}_l(i)$ that overlaps with $T_b(i)$ and the set that does not. We use operators $\Pi( )$ and $X( )$ to determine the two sets. For a given set of trees $\boldsymbol{T}_S$ and a tree T, $\Pi_i(\boldsymbol{T}_S, T)$ ($X_i(\boldsymbol{T}_S, T)$) gives a set of trees that belong to $\boldsymbol{T}_S$ and are (not) overlapped with T on R_i.

Therefore $\Pi_i(T_l(i), T_b(i))$ gives the set of trees overlapped with $T_b(i)$ while $X_i(T_l(i), T_b(i)) = T_l(i) - \Pi_i(T_l(i), T_b(i))$ gives the set of trees not overlapped with $T_b(i)$.

4. R_i then combines the forwarding entries for the trees in $\Pi_i(T_l(i), T_b(i))$ with $T_b(i)$'s entry into a single logical entry since they are overlapped on R_i.

5. R_i finally sends a $SWITCH(\Pi_i(T_l(i), T_b(i)), X_i(T_l(i), T_b(i)), T_b(i))$ message upstream on $T_b(i)$ and starts a timer for receiving possible *CONFIRM*( ) messages for this request.

It is important to highlight the fact that a member router will only consider the trees, on which it is a leaf router, as target multicast trees for tree switching. The reason is being that a successful tree switching operation will result in branch pruning. Since the protocol always starts from leaf routers, the pruning will not create routing loops or partition in a multicast tree. Another important issue in this procedure is how to determine the base multicast tree. The choice is obviously an important factor determining the overall performance of the tree switching operation and we will discuss this issue later in Section 4.5.

4.3.2 Tree Switch Procedure

By sending the *SWITCH*( ) message upstream, R_i is moved from the *Wake-up* state to the *Wait* state. We now consider an upstream router R_j on $T_b(i)$. When R_j receives the *SWITCH*( ) message, R_j moves from the *Idle* state to the *Wait* state and performs the following.

1. R_j first discovers $\Pi_j(T_l(i), T_b(i))$, the set of trees in $T_l(i)$ that overlap with $T_b(i)$ on R_j. $\Pi_i(T_l(i), T_b(i))$ and $\Pi_j(T_l(i), T_b(i))$ generally are not identical. Similar to step (4) in the wake-up procedure, R_j combines the forwarding entries for the trees in $\Pi_j(T_l(i), T_b(i))$ with $T_b(i)$'s entry into a single logical entry.

2. We denote $T_X(j)$ as the set of trees in $T(j)$ that also belong to $X_i(T_l(i), T_b(i))$. Therefore, if R_j belongs to some trees in $X_i(T_l(i), T_b(i))$, i.e., $T_X(j)$ is not empty, R_j is able to forward the traffic on $T_X(j)$ also to $T_b(i)$ by "turning on" the respective interface to $T_b(i)$. In this case R_j performs the following.

   2.1. R_j modifies, if not done so, the sets of outgoing interfaces in each entry for the trees in $T_X(j)$ so that $T_b(i)$ could also receive those traffic.

2.2. Then R_j sends a $CONFIRM(T_X(j), T_b(i))$ message back to R_i on $T_b(i)$ and it also sets $X_j(T_l(i), T_b(i))$ to $X_i(T_l(i), T_b(i)) - T_X(j)$. That is, the trees in $T_X(j)$ can be switched to $T_b(i)$.

2.3. If $X_j(T_l(i), T_b(i))$ and $\Pi_j(T_l(i), T_b(i))$ are not both empty , R_j sends a $SWITCH(\Pi_j(T_l(i), T_b(i)), X_j(T_l(i), T_b(i)), T_b(i))$ message to upstream. A timer is also set for receiving confirmation for this request. If both $X_j(T_l(i), T_b(i))$ and $\Pi_j(T_l(i), T_b(i))$ are empty, this implies that the tree switching request is completely fulfilled and there is no need to send a *SWITCH*( ) message upstream.

3. If R_j does not belong to any trees in $X_i(T_l(i), T_b(i))$, the router repeats step (2.3) with $X_j(T_l(i), T_b(i)) = X_i(T_l(i), T_b(i))$.

Based on the above, it is possible that each upstream router could partially fulfill the original tree switching request. Thus the set $X_i(T_l(i), T_b(i))$ becomes smaller in size as the *SWITCH*( ) message travels upstream. In Fig. 4.2(a), R_i will then go back and forth between the *Wait* and *Confirm* states. As long as the tree switching request is not completely satisfied, R_i will be in the *Wait* state. Similarly, the interior routers also hop between the *Wait* and *Confirm* states.

It is also possible that none of the upstream routers could satisfy the tree switching requests and the RP will be responsible for discarding the message. Consequently, the timers kept by R_i and the interior routers will expire, and they will return to the *Idle* state, as shown in both Fig. 4.2(a) and Fig. 4.2(b).

4.3.3 Confirmation Procedure

The *CONFIRM*( ) message serves two functions. The first one is to build forwarding states for the trees to be switched to $T_b(i)$. Thus when an interior router receives a $CONFIRM(T_X(j), T_b(i))$ message on $T_b(i)$, it will create a new logical forwarding entry for the trees in $T_X(j)$ and $T_b(i)$ if the trees in $T_X(j)$ are overlapped with $T_b(i)$ or for the trees in $T_X(j)$ otherwise.

The second function is to notify R_i that the tree switching request is either fully or partially satisfied. Upon receiving this message, R_i moves from the *Wait* state to the *Confirm* state and it performs the following.

1. Send a $PRUNE(T_r)$ message for each $T_r \in T_X(j)$ upstream on the tree.

2. Remove the forwarding entries for the trees in $T_X(j)$.

3. Create a new logical forwarding entry for the trees in $T_X(j)$ and $T_b(i)$ if the trees in $T_X(j)$ are overlapped with $T_b(i)$ or for the trees in $T_X(j)$ otherwise.

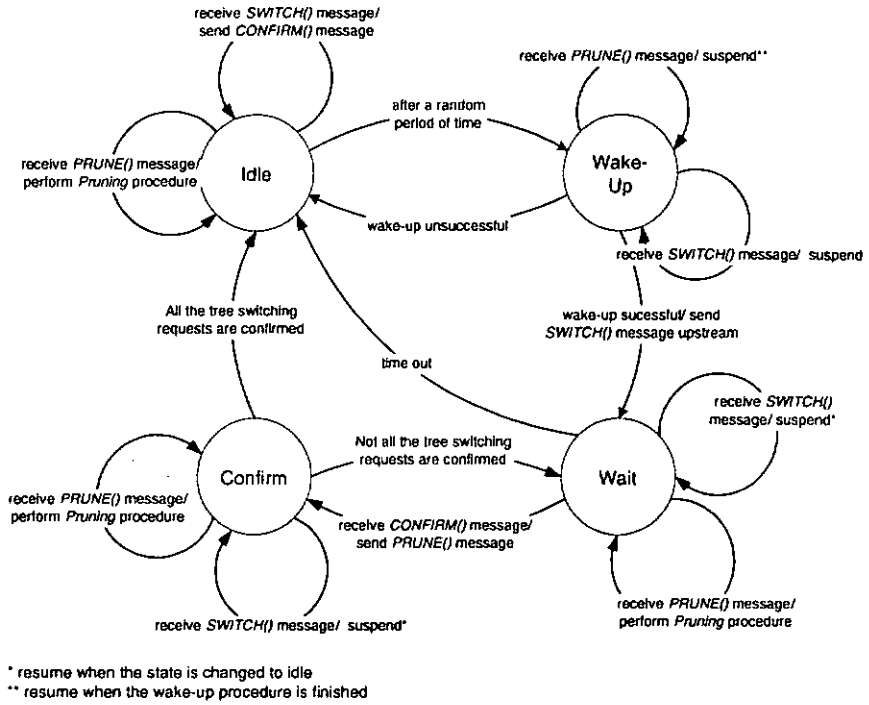
4.3.4 Pruning Procedure

The *PRUNE*( ) message is for pruning the tree branch that has already been switched to the base multicast tree. Since pruning starts from the leaf routers, the *PRUNE*( ) message travels upstream to the tree. Any routers, upon receiving the message on a tree T, are required to delete the interface, where the message is received, from the T's entry. If the resulted set of outgoing interfaces is empty and the router is an on-tree router, the forwarding entry will be removed and the *PRUNE*( ) message will be forwarded further upstream.

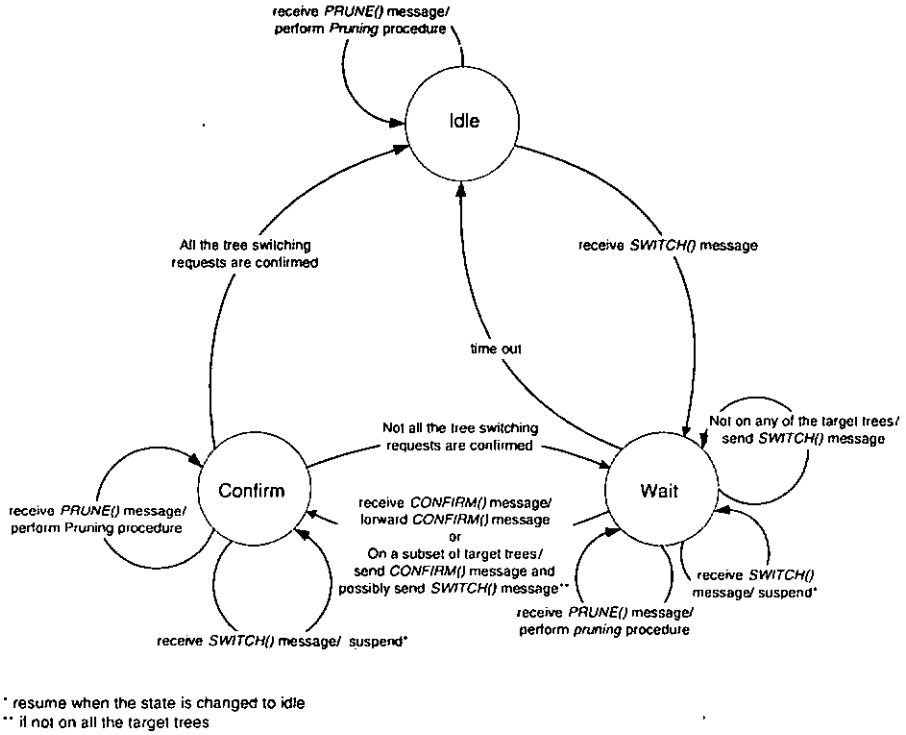
4.3.5 Concurrency Control

Since it is possible to have multiple tree switching operations ongoing at the same time, it is important to make sure that routing loops and tree partitions will not occur during tree switching process. In Fig. 4.2(a), we indicate that it is possible to receive *SWITCH*( ) and *PRUNE*( ) messages when a router is in any one of the four states. A router is required to immediately execute pruning requests when it is in the *Idle*, *Wait* or *Confirm* states. The pruning does not affect the TSP's operation since $T_b(i)$ and the trees in $T_l(i)$ will not be pruned away by the message. First, R_i will not receive *PRUNE*( ) messages for $T_l(i)$, the set of target multicast trees, since R_i is a leaf router on the trees in $T_l(i)$. Second, R_i will stop forwarding the *PRUNE*($T_b(i)$) message upstream since it is a designated router. However, the router will suspend any *PRUNE*( ) message received when it is in the *Wake-up* state until it leaves the state. In the *Wake-up* state, the router is determining or has determined the $T_l(i)$ and the pruning procedure may turn an interior router into a leaf router, thus making $T_l(i)$ inconsistent. The pruning procedure will be resumed when the router finishes processing the wake-up procedure.

On the other hand, a router will suspend *SWITCH*( ) messages received during the *Wake-up*, *Wait* or *Confirm* states until it returns to the *Idle* state. A *SWITCH*( ) message may turn a leaf router into an interior router because the leaf router may extend itself to the downstream routers to satisfy the tree switching request. At the same time, this router, upon receiving a *CONFIRM*( ) message in response to its earlier tree switching request, will perform pruning on the original tree branch. Since this router is no longer a leaf router, the tree pruning will result in tree partitions.



(a) State transition diagram for member routers.



(b) State transition diagram for interior routers for switching a set of target trees.

Fig. 4.2. TSP's State transition diagram.

4.3.6 Tree Join and Leave

The discussion on the TSP's operation so far assumes that the multicast trees under consideration do not change throughout the protocol operation. Nevertheless, multicast trees do change in size and shape in the real-world environment. With a new router added to a multicast tree, a member router may no longer be a leaf router and problems may arise when TSP performs pruning later on. To handle this situation, a member router, which initiated a tree switching request and the process has not been completed yet, will suspend any tree join requests until all the outstanding tree switching processes are completed. On the other hand, the leaving of a member router will affect on-going tree switching processes initiated by this router. Similar to the case of tree join, any tree leave requests from a member router will be delayed until all the outstanding tree switching requests initiated by this router are completed.

4.3.7 Synchronization Between Unicast and Multicast Paths

One important consequence of the tree switching operation is the asynchronization between unicast routing path and multicast routing path. In PIM-SM, each member router builds a tree branch to the RP using shortest path. Therefore, a router will use the same interface to send unicast packets to the RP and to receive multicast packets forwarded by the same RP. Instead of using the shortest path, TSP attempts to find an alternative path from a member router to the RP so as to reduce the number of routers lie on and hence the number of forwarding entries maintained at the multicast trees. As a result, the paths provided by TSP may be different from those used in PIM-SM.

Since we apply TSP on top of PIM-SM, we have to deal with the periodic refresh performed in PIM-SM. To adapt to the dynamics of underlying network, PIM-SM requires each member router to send a *Join/Prune* message toward the RP periodically. Since PIM-SM uses unicast routing table to perform periodic refresh, the routers lying on the tree branches switched by TSP may not receive a *Join/Prune* message before timeout. As a result, the entries maintained in those routers are expired and then deleted. To reduce multicast forwarding state, TSP is required to perform again. If the interval between two refreshes is small, the tree branches will be built and deleted frequently. Since tree switching involves message passing along the multicast tree, it may introduce a lot of message passing in the network if the tree switching is performed too frequently.

To justify the overhead of tree switching, we propose to increase the interval between two refreshes. Even though the increase in timer interval will deteriorate the adaptability to the network dynamics, the performance of PIM-SM will not be affected significantly if the number of network changes is small or the time interval between two changes is long.

Another implication of the asynchronous issue is that TSP assumes a separate multicast routing table. TSP therefore works with any "protocol independent" multicast routing protocols

that employ a separate multicast routing table like PIM-SM, PIM-DM, CBT, OCBT, and MIP. But TSP cannot work with DVMRP which does not have a separate multicast routing table but instead it associates multicast routing information with the unicast routing table.

4.4 Properties of TSP

In this section, we show that TSP is routing loop free and incurs no packet loss. These two properties are in fact the requirements when designing TSP.

Property 1: TSP is routing loop free.

Proof: Since each router makes use of a dedicated interface to connect to a LAN, an interior router on a multicast tree must have more than one incoming interface in order to have a routing loop. However, this is not possible with TSP. The only time that TSP will add a new incoming interface is when a tree switching request is successful. However, the protocol will prune away the original incoming interface as soon as it receives the *CONFIRM*( ) message. Furthermore, the tree switching is performed for a forest of shared trees or a forest of source-specific trees. Tree switching is not allowed for a forest of mixed trees in order to avoid possible routing loops.

Property 2: There is no loss of data packet as a result of performing TSP.

Proof: Data loss with TSP is not possible since the designated router that initiates the protocol will not prune away the original path before receiving the *CONFIRM*( ) message. Also, by the time the *CONFIRM*( ) message is received, a new path is already set up. Moreover, the pruning always takes place from the leaf routers and therefore pruning will not create tree partitions.

4.5 Simulation Results

In our experiments, we represent network topologies in the form of random graphs. These random graphs are generated using the algorithms described in [Waxman 1988]. A random graph, N, is denoted as $N = (V, E)$ where V is a set of nodes which represents multicast routers and E is a set of edges which represents network links. These nodes are randomly distributed over a Cartesian coordinate system. The probability that an edge exists between any two nodes u and v is given by the probability function, $P(\{u, v\})$, which is, in turn, defined as

$$P(\{u,v\}) = \beta \times e^{\frac{-d(u,v)}{L\alpha}} \tag{4.1}$$

where $d(u, v)$ is the distance from u to v, L is the maximum distance between any two nodes, and α and β are parameters in the range (0, 1]. The density of short edges relative to longer ones decreases as α increases whereas the edge densities increases as β increases [Waxman 1988].

In our simulation experiments, we set the average nodal degree to four and the cost of each link is assumed to be unity. We then generate a random graph consists of 100 nodes. Our simulations are performed in random networks with group density of 0.05 and 0.2. The group density (GD) of a multicast group is defined as the fraction of routers that are member routers in the multicast group. We select the member routers and RP randomly from the graph for each multicast tree. The number of multicast groups is varied form 5 to 95 for each group density. By selecting a shortest path from a member router to the RP, a shared tree is thus formed. Each data point in our simulation results is an average over 100 independent simulation experiments. In addition to the averages, we also measure the 95% confidence interval of each data point. The 95% confidence interval is shown in the form of error bars for each data point in the figures. When a member router R_i starts the TSP, it selects the base multicast tree randomly from the set $T(i) - T_I(i)$ if the set is not empty; otherwise it selects randomly from $T_I(i)$.

In this section, we will evaluate the performance of TSP when applied to PIM-SM. We will only consider the case of shared trees. The source-specific trees (e.g DVMRP, MOSPF, and PIM-DM) will not be considered as they can be treated similarly. Specifically, we will use simulation to evaluate the effect of TSP in the areas of tree overlapping, state reduction, and packet delay in terms of hop counts. In addition, we will evaluate the impact of base tree selection method on the aforementioned performance areas. Lastly, we will examine the performance of TSP in dealing with group dynamics.

4.5.1 Degree of Tree Overlapping

We first define a metric for measuring the *degree of tree overlapping* for a set of shared trees $\boldsymbol{T}$ in (4.2).

$$DTO(\boldsymbol{T}) = \frac{\sum_{\text{all } L_i \in \boldsymbol{L}} \left(\frac{n_i - 1}{n_i} \times \frac{N(\boldsymbol{T})}{N(\boldsymbol{T}) - 1} \right)}{N(\boldsymbol{L})} \tag{4.2}$$

where $DTO(\boldsymbol{T})$ is the degree of tree overlapping for $\boldsymbol{T}$, which ranges from 0 to 1. $\boldsymbol{L}$ is the set of links that are on at least a tree in $\boldsymbol{T}$. $N(\boldsymbol{T})$ and $N(\boldsymbol{L})$ are the total number of multicast trees in $\boldsymbol{T}$ and the total number of links in $\boldsymbol{L}$ respectively. For a link L_i in $\boldsymbol{L}$, n_i is the number of multicast trees that this link belongs to.

It is important to note that L_i does not contribute to the $DTO(\boldsymbol{T})$ if the multicast trees are not overlapped on L_i. On the other hand, if L_i is on all the multicast trees in $\boldsymbol{T}$ and they are all overlapped with each other, L_i contributes $\frac{1}{N(\boldsymbol{L})}$ to the $DTO(\boldsymbol{T})$. In a special case, where all the multicast trees in $\boldsymbol{T}$ are completely overlapped, $DTO(\boldsymbol{T})$ is equal to 1.

We show the *DTO* under two cases in Fig. 4.3, where the group densities are given by 0.05 and 0.2, and the number of multicast trees ranges from 5 to 95.

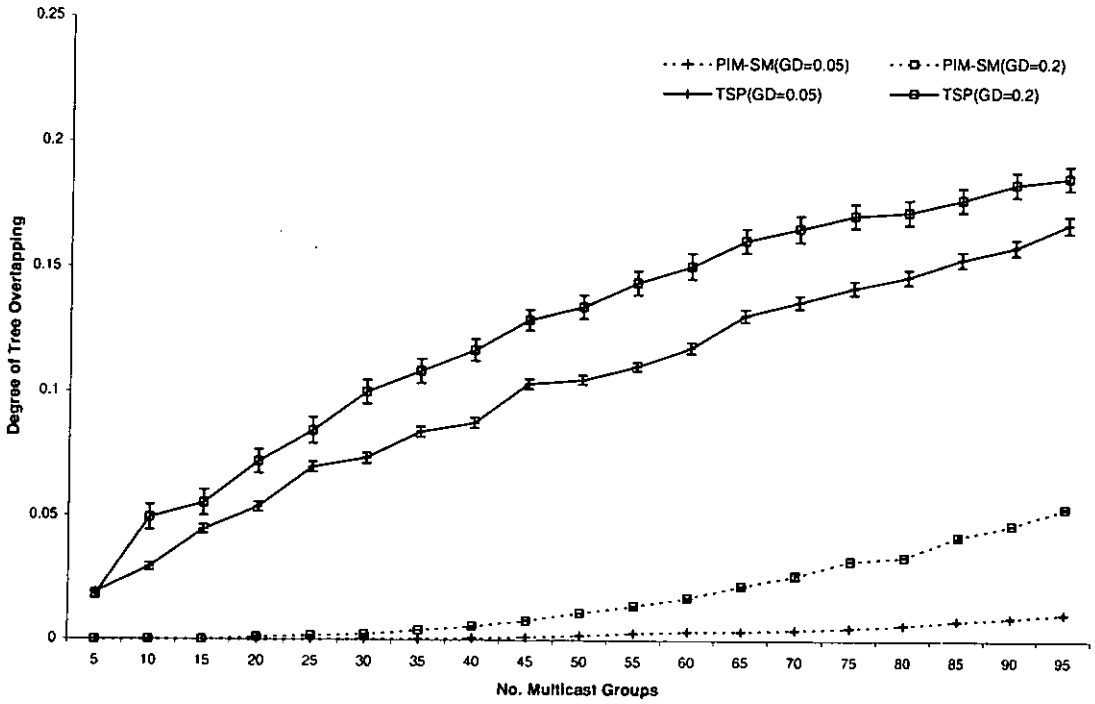


Fig. 4.3. TSP and degree of tree overlapping.

Let us consider the degree of tree overlapping in the two cases without TSP at first. Since there are more multicast trees overlapped on a link, it is more likely that a branch of a multicast tree overlaps with branches of others when the number of multicast trees increases. As a result, the *DTO* increases with the number of multicast trees. With a higher group density, the rate of increase in *DTO* is higher since there is, in general, a higher probability for the trees to be overlapped on a router.

When we apply TSP to both cases, the *DTO* increases even further. Specifically, the *DTO* for the group density of 0.2 is greater than that for the group density of 0.05 because it is more likely that the multicast trees are overlapped on a router with higher group density. Nevertheless, the rate of increase in the *DTO* for the group density of 0.2 is less than that for the group density of 0.05 especially when the number of multicast trees is large. It is because of the fact that the original forest of multicast trees already has a significant overlapping among themselves and therefore little is gained by applying TSP.

4.5.2 Multicast State Reduction

We measure the state reduction according to (4.3).

$$SR(T) = \frac{E_P(T) - E_S(T) \times AR}{E(T)} \tag{4.3}$$

where $SR(T)$ is the percentage of state reduction for a set of multicast trees T. $E(T)$ is the average total number of multicast forwarding entries required for the multicast trees in T. After executing the TSP on T, $E_P(T)$ is the number of multicast forwarding entries pruned away and $E_S(T)$ is the number of new multicast forwarding entries added to the base multicast trees, as part of logical entries. Since the switched tree branches must be overlapped with at least the base tree, the states for those entries can be further reduced by performing state aggregation as explained earlier. Now we denote the aggregation ratio as AR and we consider AR = 0.5, 0.8, and 1 in Fig. 4.3. The storage for each new entry added to the base multicast trees is therefore discounted by AR. The case of AR = 1 is equivalent to no state aggregation at all. Therefore, the amount of state reduction in this case is obtained by the difference between $E_P(T)$ and $E_S(T)$.

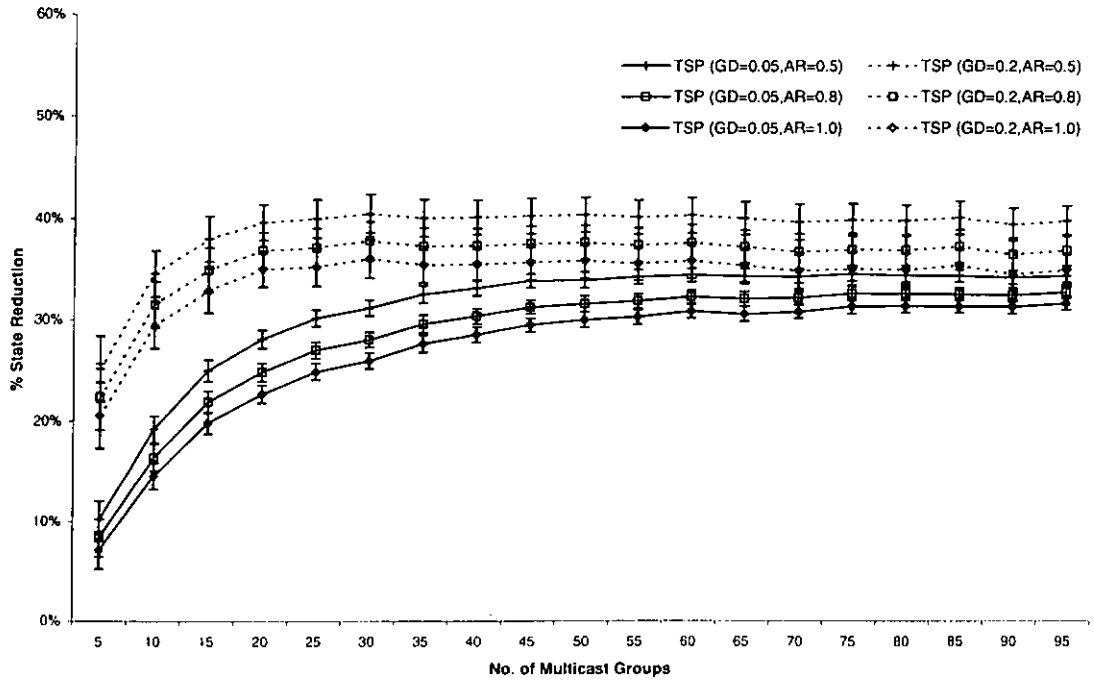


Fig. 4.4. TSP and multicast state reduction.

As shown in Fig. 4.4, the TSP is able to reduce about 31% of the multicast states for the group density of 0.05 even without state aggregation for overlapped multicast trees. With state aggregation, the total state reduction can be increased by an additional 1-3%. On the other hand, the TSP is able to reduce about 35% of the multicast states for the group density of 0.2 without state aggregation for overlapped multicast trees. With the state aggregation, the states can be reduced by an additional 2-5%, a wider margin than the case of 0.05.

It is important to note that the state reduction for the group density of 0.2 is approaching a plateau earlier than that for the group density of 0.05. Since more multicast trees are overlapped at a router with higher group density, the condition that the forest of multicast trees already has a significant overlapping among themselves and little is gained by applying TSP is reached earlier in the case of 0.2 than the case of 0.05. It is for this reason that the curve for the group density of 0.2 becomes flattened earlier than that for the group density of 0.05.

Furthermore, the state reduction for a sparse shared tree (group density of 0.05) is mainly gained from switching trees to the base multicast trees. The average "length" of a pruned tree branch is longer than the average "length" of a tree branch switched to a base tree because the number of on-tree routers between a leaf router and the closest member router upstream is higher for a sparse tree. Consequently, more on-tree routers will be pruned away after tree switching and the additional gain from compressing the states for overlapped trees (1-3%) is not very large.

On the other hand, when the shared trees are more bushy (group density of 0.2), the contribution from tree switching becomes less but that from the state aggregation for overlapped trees increases. This is because the number of on-tree routers to be pruned away is less when there are more member routers on the multicast tree. In addition, there is already a large degree of tree overlapping when the group density is high. It is for this reason that state aggregation (2-5%) makes a more significant contribution to the state reduction in this case.

4.5.3 End-to-End Delay

TSP has an adverse effect on the end-to-end delay in terms of hop counts. For each multicast tree, we randomly pick 5 nodes in the network as the designated routers for source hosts (first-hop routers) and we measure the average hop count from the first-hop routers to other member routers. The first-hop routers will send packets to the RP first and then the RP will forward them to other member routers. Fig. 4.5 shows the effect of TSP on average end-to-end delay.

In both cases, we observe an average increase of 1 hop in the end-to-end delay and the average end-to-end delay without tree switching is 7 hops. As shown in Fig. 4.5, the increase is not sensitive to the number of multicast groups and the group density. The tree switching will likely move the member routers further away from the RP and hence provide a longer path from the RP to the member routers. As a result, TSP introduces an increase in hop count.

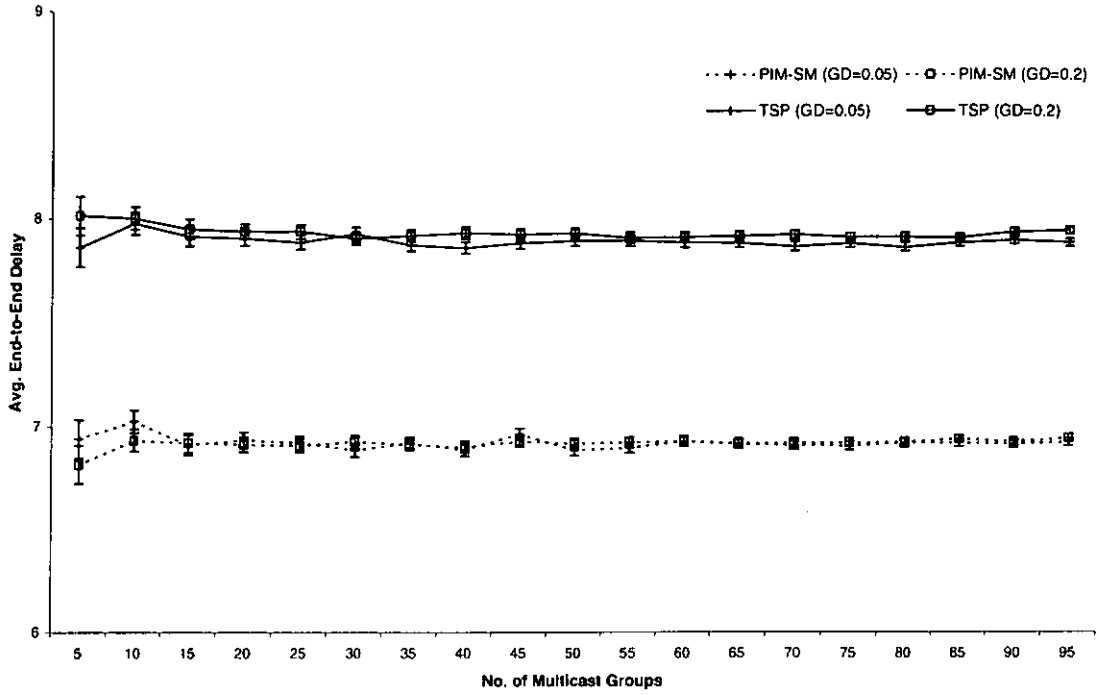


Fig. 4.5. TSP and end-to-end delay (hop count).

4.5.4 Base Multicast Tree Selection

So far, we select the base trees randomly. In this section we will consider another selection method that is based on a static criterion such as the multicast group addresses. We call this method *static selection* in contrast to the random selection employed so far. Whenever a member router selects a base tree, the selection will be based on the smallest multicast address. Fig. 4.6 shows the difference of the two selection methods in the degree of tree overlapping.

In Fig. 4.6, we observe that the random selection method gives a higher degree of tree overlapping than the static method. In particular, random selection method is able to increase the degree of overlapping by 9% in average when group density is 0.05 and 15% in average when group density is 0.2 when compare to the static method. The reason for this is that the static selection method always restricts the base tree to the one with the smallest address. If the corresponding tree does not "cross-over" with the target trees upstream, the tree switching would not be successful. Unlike the static method, the random selection method will explore other trees if the previous wake-up is unsuccessful. As a result, the probability of a successful tree switching is higher with the random selection method and it results in more tree overlapping.

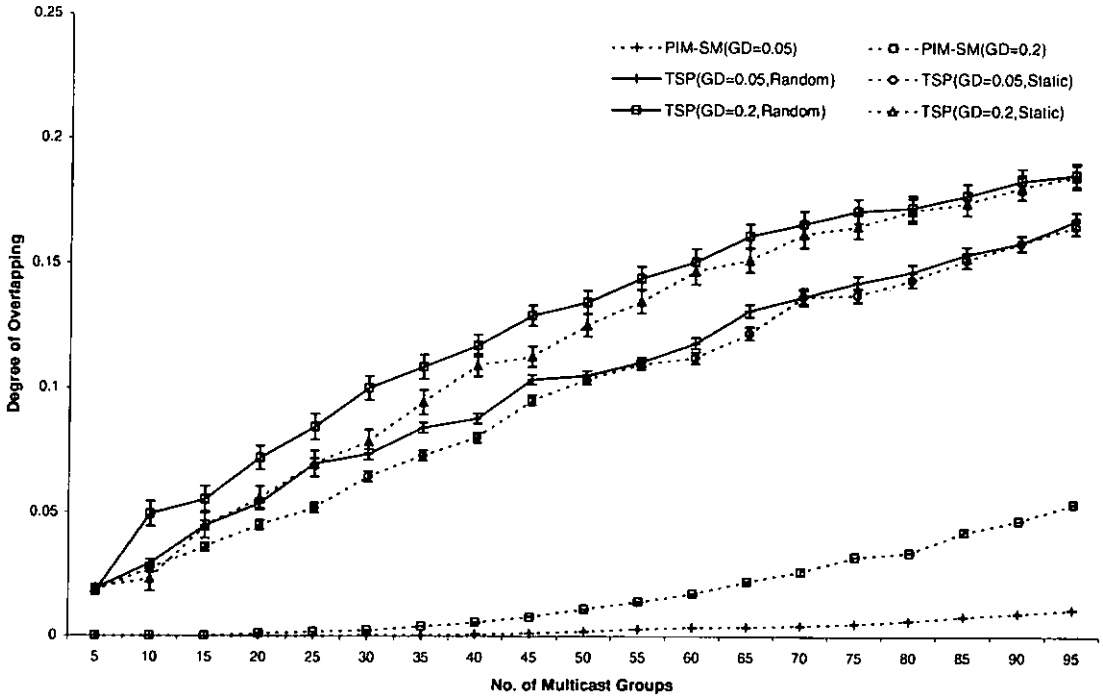


Fig. 4.6. Base multicast tree selection method and degree of tree overlapping.

In Fig. 4.7 and Fig. 4.8 below, we compare the two selection methods using the percentage of state reduction.

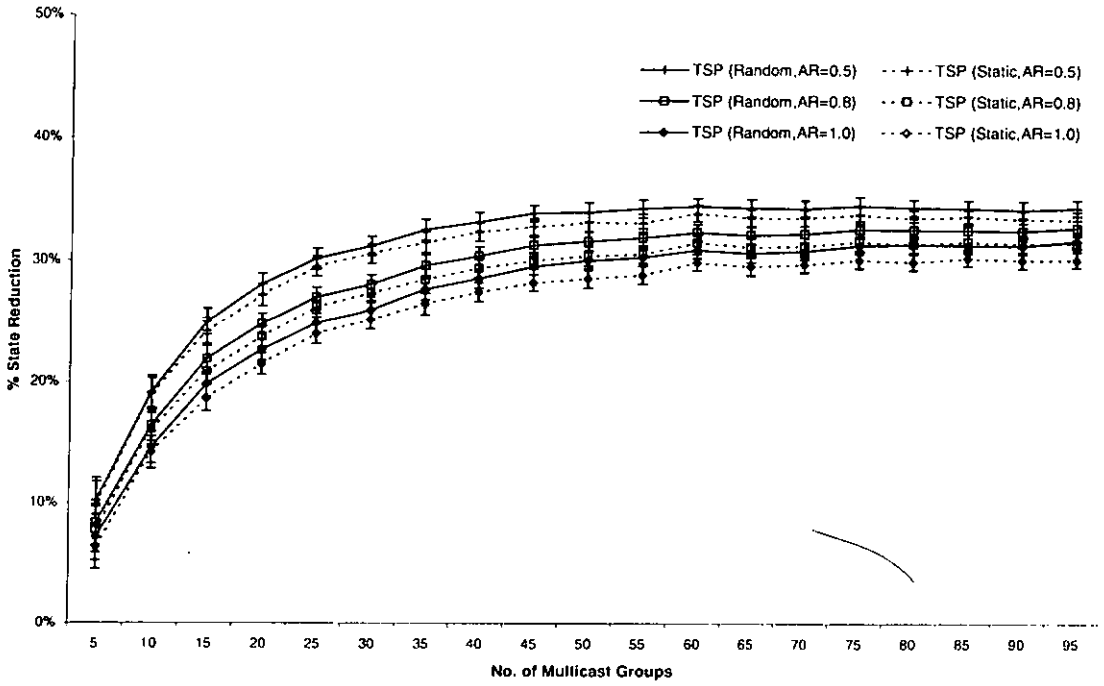


Fig. 4.7. Base multicast tree selection & state reduction (group density = 0.05).

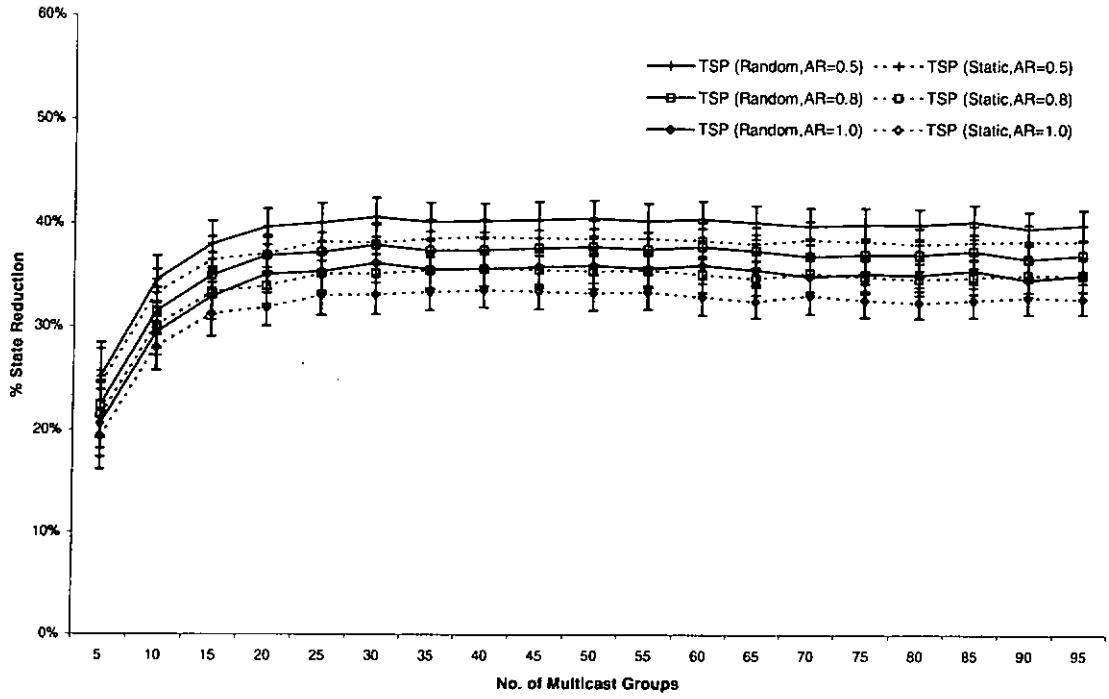


Fig. 4.8. Base multicast tree selection & state reduction (group density = 0.2).

In both group densities, the random selection method gives a higher state reduction than the static method with different *AR*. This is consistent with the previous results in the degree of tree overlapping. The reason is that there are more multicast trees for the random method to select when the number of multicast trees is large. This allows the random selection method to have greater chance to switch the tree successfully. On the other hand, the static method can only select one multicast tree no matter how many multicast trees are there. As a result, the increase in the number of multicast trees does not affect the probability of a successful tree switching in the static selection method.

Let us further note that the superiority of the random selection method to the static method in terms of state reduction becomes more significant in the case of 0.2 than that in the case of 0.05. It is because of the fact that more multicast trees are overlapped at a router with higher group density. As a result, the random selection method is able to select the base multicast tree from a larger number of trees and therefore the tree switching can be successfully completed with higher probability. Unlike random selection method, the static method can only select one multicast tree regardless of the total number of multicast trees the router lies on. This makes the probability of a successful tree switching to be independent of the number of multicast trees.

4.5.5 Overhead of Control Messages

The control overhead of TSP is measured in terms of the number of control messages traversed in each link. In particular, we measure the number of *SWITCH*( ), *CONFIRM*( ), and *PRUNE*( ) messages in the course of tree switching. Fig. 4.8 shows the control overheads of TSP with the group density of 0.05 and 0.2 respectively.

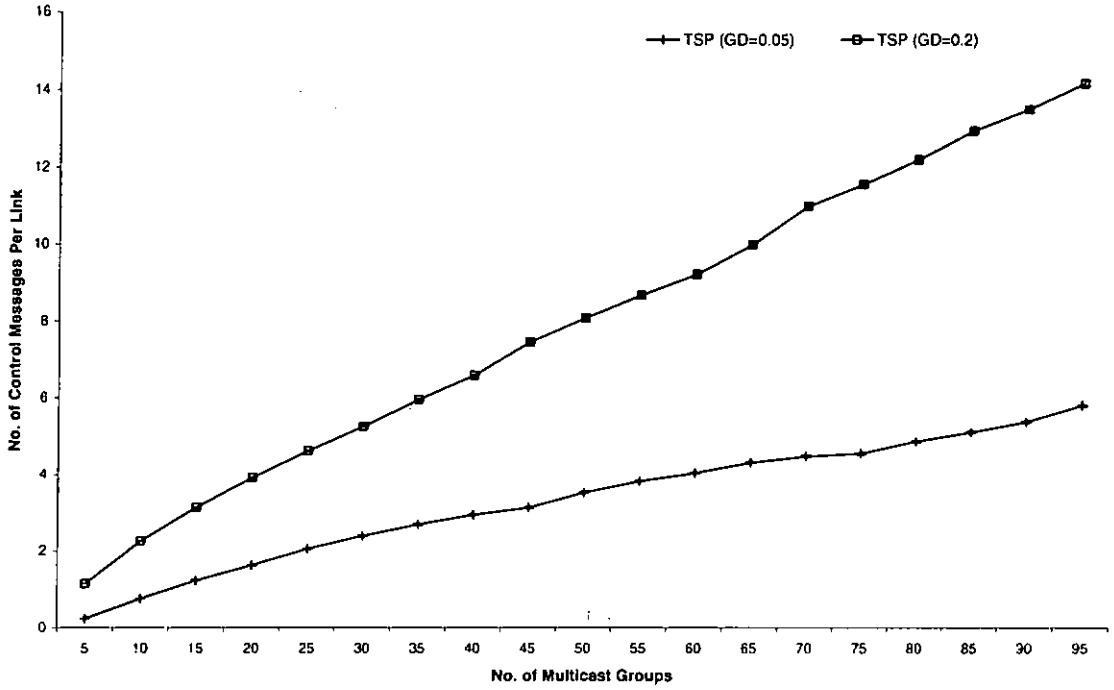


Fig. 4.9. The control overhead of TSP.

Since the amount of tree switching operations is proportional to the number of multicast trees, the control overhead of TSP increases as the number of multicast trees increases (Fig. 4.9). Furthermore, the control overhead of TSP is greater with higher group density when the cases with the group density of 0.05 and 0.2 are taken into consideration. Since there are more and more leaf routers and hence the probability that a member router wakes up successfully increases as group density increases, the amount of tree switching operations increases. It is for this reason that the control overhead of TSP is greater with higher group density.

4.5.6 Adaptability to Membership Dynamics

To evaluate the performance of TSP under dynamic changes of membership in the Internet applications over time, we run simulations in which members join/leave a multicast group over time according to the Poisson distribution. The Poisson probability distribution is expressed as

$$f(x) = \frac{\lambda^x e^{-\lambda}}{x!} \tag{4.4}$$

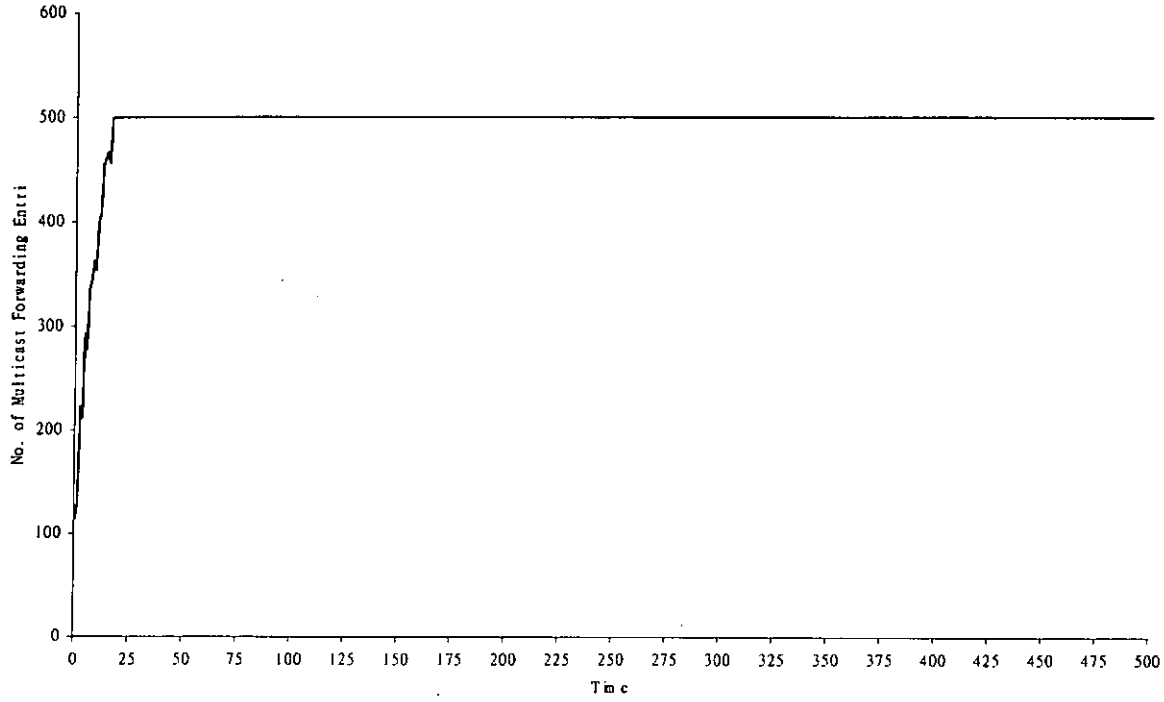
In our simulations, it is possible that a new member joins the multicast group and an existing member leaves the multicast group. The above events are occurred according to the Poisson distribution with different value of λ. We denote the rate for a new member to join the multicast group as λ_1 and the rate for an existing member leaves the multicast group as λ_2.

To demonstrate how membership dynamics affect the performance of TSP, we vary the rate for a new member to join and the rate for an existing member to leave but we keep the number of multicast groups and the number of sources constant. In the simulation experiments, we use two settings of the rates: (i) $\lambda_1 = 8$, $\lambda_2 = 2$; and (ii) $\lambda_1 = 8$, $\lambda_2 = 4$. We set the number of multicast groups to 5 and the number of sources in each multicast group to 5. We assume the multicast sessions will not finish once they have started in our simulations since multicast sessions in intra-domain and inter-domain levels tend to be long-lived. Let us note that we fix the number of multicast groups in these simulations because the performance of TSP is affected by the number of multicast groups. The way that TSP performs as the number of multicast groups varies has already been evaluated in the previous sections, that is, Section 4.5.1-4.5.5. On the other hand, the reason why we fix the number of sources in each multicast group is that the performance of a shared tree is independent to the number of sources.

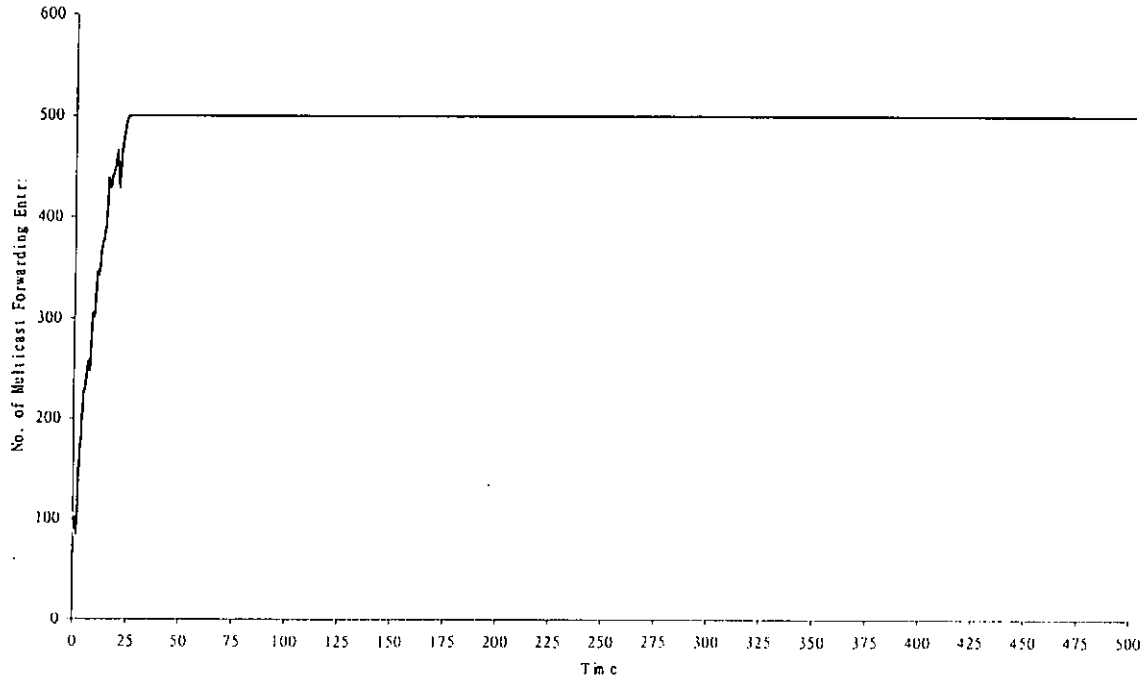
We then measure the performance of TSP in terms of scalability, average end-to-end delay, and control overhead in every time unit. The scalability is the total number of multicast forwarding entries for all the multicast distribution trees after performing TSP without applying any state aggregation. The average end-to-end delay is the average number of hops from first-hop routers to member routers. The control overhead is the number of control messages per link generated by TSP in the tree switching process. Each member router maintains a timer to perform periodic refresh every 60 time units. We run the simulations over 500 time units.

Fig. 4.10 shows the scalability of TSP in terms of number of multicast forwarding entries maintained at on-tree routers over time. As shown in Fig. 4.10, the number of multicast forwarding entries increases and then stabilizes at certain threshold. It is because the rate of member joins the multicast group is greater than that of member leaves. As a result, there is a net increase in the number of on-tree routers. Since each on-tree router has to maintain a multicast forwarding entry for each multicast group, the number of entries increases as the number of on-tree routers increases. As time goes by, almost all the routers in the network become member routers. As a consequent, the number of multicast forwarding entries stabilizes at certain threshold. The number for multicast forwarding entries reaches the threshold in Fig. 4.10(a) at time unit smaller

than that in Fig. 4.10(b) because the rate of member leaves in Fig. 4.10(a) is slower than that in Fig. 4.10(b).



(a) $\lambda_1 = 8, \lambda_2 = 2$



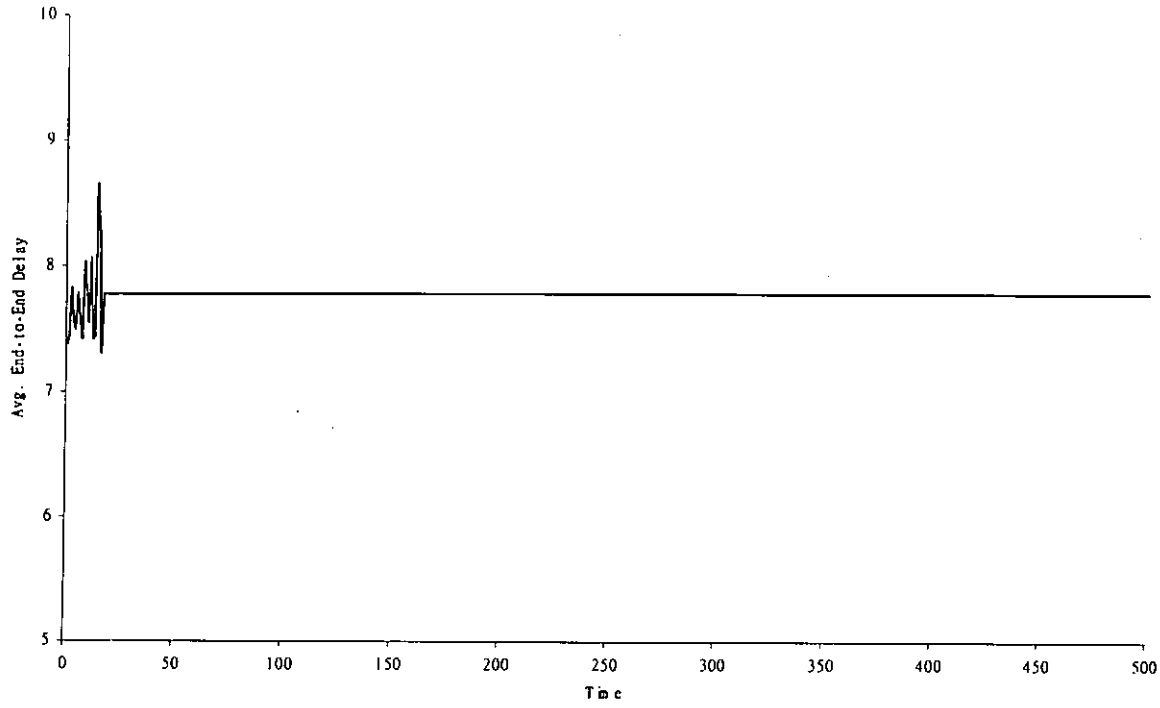
(b) $\lambda_1 = 8, \lambda_2 = 4$

Fig. 4.10. Scalability of TSP over time.

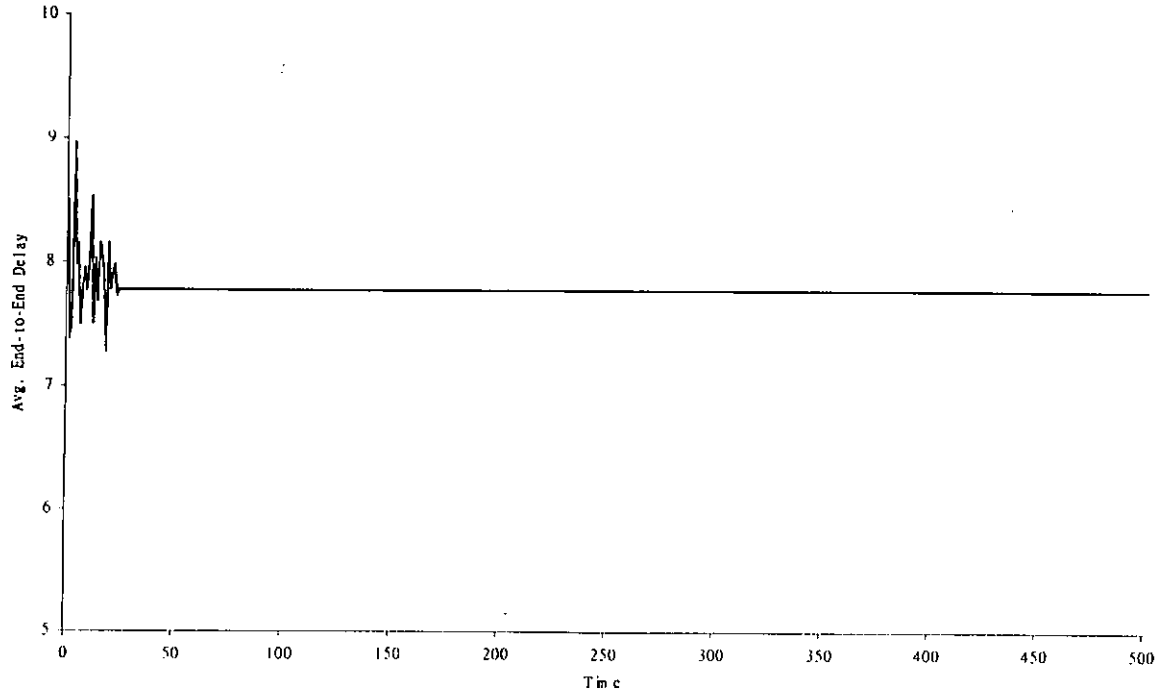
The average end-to-end delay of TSP over time is given in Fig. 4.11. The readers should note that the average end-to-end delay increases/decreases sharply and then becomes more or less the same when time unit is greater than 25. The sudden changes in average end-to-end delay are due to the joins of members to and leaves of members from the multicast group. Afterwards, the joins and leaves of members have little effect on the average end-to-end delay because almost all routers are already on-tree no matter how the group memberships are changed. As shown in Fig. 4.11, the average end-to-end delay changes more frequently in Fig. 4.11(b) than that in Fig. 4.11(a) when the time unit is smaller than 25 because the rate of member leaves in Fig. 4.11(b) is faster than that in Fig. 4.11(a).

The control overhead of TSP over time is given in Fig. 4.12. Similar to the average end-to-end delay, the control overhead changes significantly because members join and leave the multicast group frequently when the time unit is small (≤ 25). Since members join and leave the multicast group affect the probability that a member router wakes up successfully and hence the amount of tree switching operations, the control overhead changes significantly as members join and leave the multicast group frequently. As time goes by, the joins and leaves of members have little effect on the control overhead because almost all routers are already on-tree no matter how the group memberships are changed. Consequently, the control overhead is comparatively small when time unit is greater than 25.

It is also important to note that we assume each member router performs periodic refresh every 60 time units. Since the paths provided by TSP may be different from those used by the underlying multicast routing protocols, the refresh messages, which re-builds the branches of multicast distribution trees constructed by the underlying multicast routing protocol, may not traverse the branches switched by TSP. These branches are therefore removed. As a result, each member router performs tree switching every 60 time units. Since member routers join multicast groups at different time units, their timers expire at different time units. As a result, there are some member routers perform periodic refresh at each time unit. Consequently, there are some control messages generated and processed in each time unit. As shown in Fig. 4.12, the control overhead changes more frequently in Fig. 4.12(b) than that in Fig. 4.12(a) when the time unit is small because the rate of member leaves in Fig 4.12(b) is faster than that in Fig. 4.12(a).

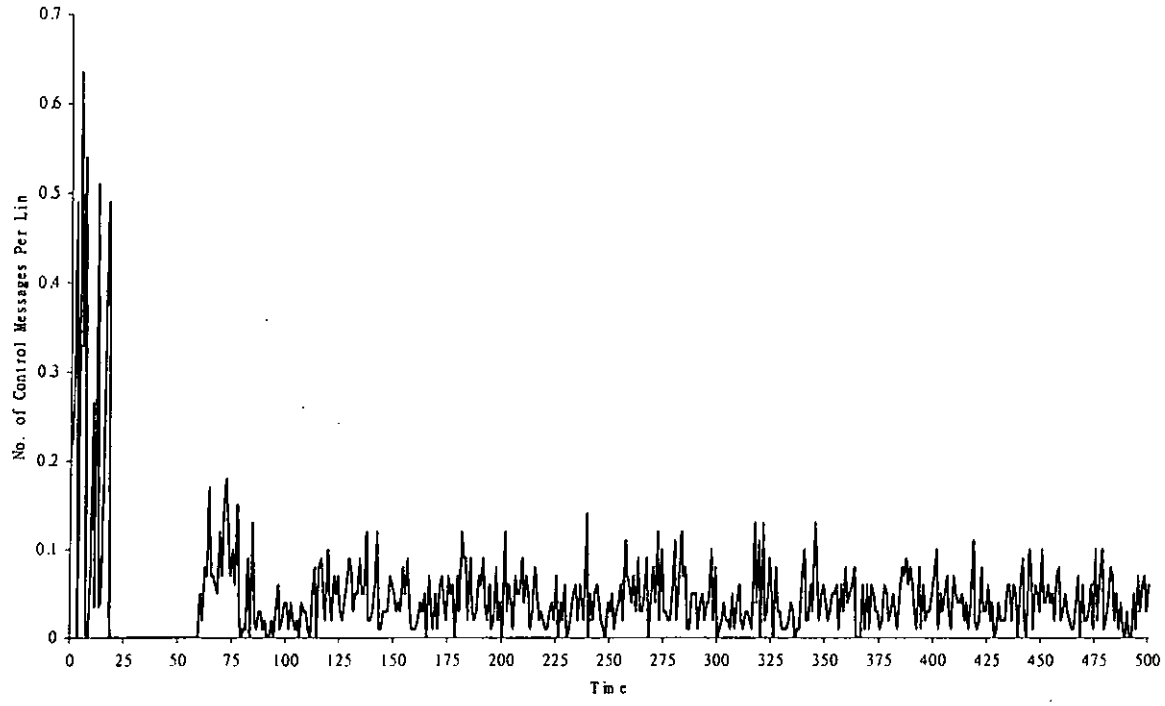


(a) $\lambda_1 = 8, \lambda_2 = 2$

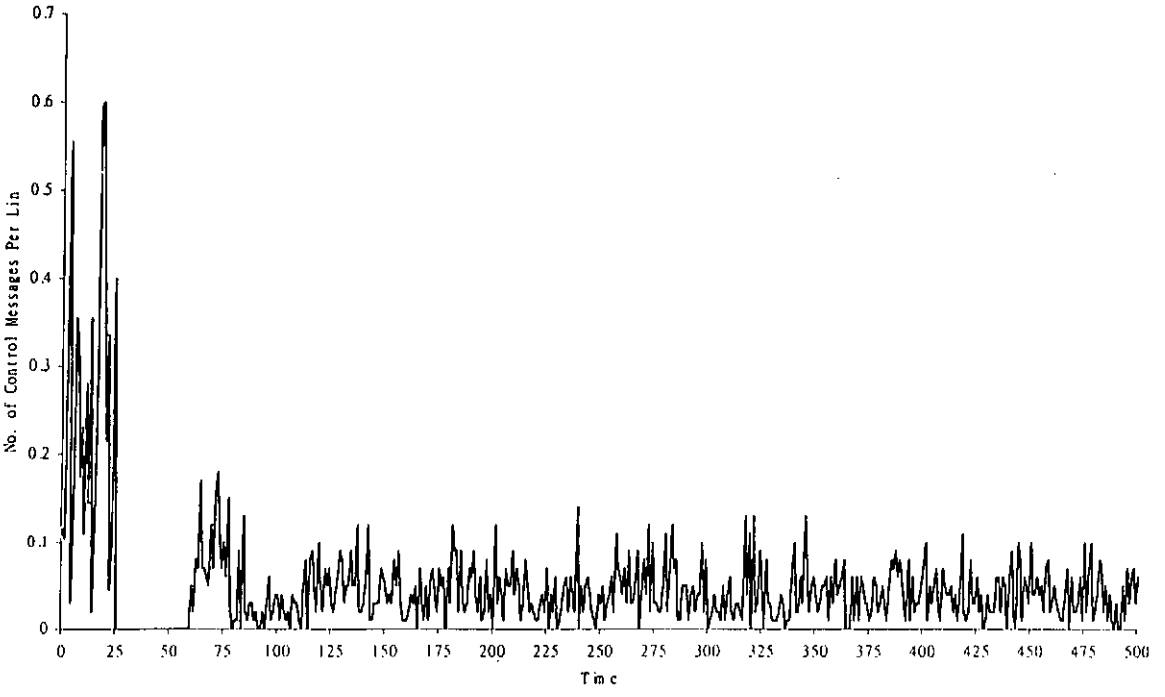


(b) $\lambda_1 = 8, \lambda_2 = 4$

Fig. 4.11. Average end-to-end delay of TSP over time.



(a) $\lambda_1 = 8$, $\lambda_2 = 2$



(b) $\lambda_1 = 8$, $\lambda_2 = 4$

Fig. 4.12. Control overhead of TSP over time.

4.7 Summary

In this chapter, we have proposed a new Tree Switching Protocol (TSP) for reducing the total state requirement for a forest of shared trees or a forest of source-specific trees. This protocol exploits overlapping among multicast trees by switching multicast trees to a base multicast tree. Simulation results show that a significant state reduction could be obtained for a sparse tree through tree switching. If state aggregation for overlapped trees is performed, a further state reduction can be obtained but the amount is not significant for sparse trees. TSP is a very efficient distributed protocol. A careful concurrency control is implemented into the protocol so that it is routing loop free and there is no packet loss.

Chapter 5

Conclusions and Future Work

Existing multicast routing protocols are based on either source-specific trees or shared trees. The protocols based on source-specific trees (e.g. DVMRP, MOSPF) aim at achieving efficient end-to-end delay whereas the protocols based on shared trees (e.g. CBT, PIM-SM, MIP, and OCBT) aim at achieving good scalability to the number of sources in a multicast group. Regardless of whether an existing multicast routing protocol considered is based on source-specific or shared trees, it is developed to achieve either good scalability or efficient end-to-end delay but not both of them. This prevents existing multicast routing protocols from supporting those Internet applications, which require efficient distribution of packets over many sources in a multicast group.

In order to achieve good scalability to the number of sources in a multicast group and efficient end-to-end delay at the same time, we propose to use a novel multicast routing protocol called Adaptive Source and CORe Based MulticasT (ASCORT) in this thesis. The multicast distribution tree constructed by ASCORT is composed on a forest of reduced trees and a bi-directional tree. A reduced tree, which is rooted at one of the sources, connects to a subset of group members whereas a bi-directional tree, which is rooted at the first-hop router of a source, connects to the first-hop routers of all the other sources in the multicast group. The group members connecting to a reduced tree is able to experience optimal end-to-end delay from the root of the reduced tree. The construction of reduced trees effectively partitions group members into different disjoint subsets. Each router lying on the multicast distribution tree is therefore required to maintain one forwarding entry only. Furthermore, the use of bi-directional tree to connect the first-hop routers of sources together makes ASCORT become adaptive to the location of sources and it is for this reason that ASCORT is able to achieve efficient end-to-end delay. The simulation results show that ASCORT is able to achieve better scalability and more efficient end-to-end delay than PIM-SM. Together with the fact that ASCORT is a distributed protocol, routing loop free, and very efficient, ASCORT is applicable to many large-scale Internet applications in the real world.

It is also important to note that existing multicast routing protocols are inadequate to scale to the number of multicast groups no matter they are based on source-specific trees or shared trees. To tackle this problem in an effective manner, we introduce an innovative technique called Tree Switching Protocol (TSP) in this thesis. TSP, which is a distributed protocol running on top of any unicast protocol independent multicast routing protocol, is able to reduce multicast forwarding

states required for a forest of shared trees or a forest of source-specific trees. The protocol accomplishes this goal by allowing each member router to select a base multicast tree and "switch" other multicast trees to the base tree. If the multicast trees are not overlapped with the base tree, the tree switching operation will result in more overlapping among the multicast trees and a net reduction in multicast state. The simulation results show that the tree switching operation is able to reduce multicast forwarding states by a significant amount. Together with the fact that it is loop-free and very efficient, TSP can be applied to large-scale Internet applications.

As shown in Chapter 4, TSP alone is able to reduce multicast forwarding states maintained at multicast routers by a substantial amount. The amount of forwarding states can be reduced further by means of multicast state aggregation. In the future, we will develop multicast state aggregation methods so as to increase the reduction in forwarding states maintained at multicast routers. By aggregating multicast state, multicast routing protocols are able to better scale to the number of multicast groups.

In addition to introducing a new state aggregation method, we will also study the applicability of TSP to Multiprotocol Label Switching (MPLS) networks. MPLS is an emerging standard technology that offers new capabilities for large-scale IP networks. It is able to improve the price/performance of network layer routing and the scalability of the network layer. Furthermore, MPLS allows new routing services to be provided without making any change to the forwarding paradigm. MPLS is therefore superior to any existing IP technology in enabling several large-scale applications such as traffic engineering (the ability of network operators to dictate the path that traffic takes through their network) and Virtual Private Networks (VPNs) support. Nowadays, MPLS has applications in the deployment of IP networks across ATM-based wide area networks, in providing traffic engineering capabilities to packet-based networks, in providing IP QoS capabilities, and in aiding the deployment of IP-based VPNs.

In order to apply TSP to MPLS networks, we will introduce a method to identify the overlapping branches of a set of trees by the same label. Given a set of trees, it is common that some of their branches are overlapping with each other. Instead of using a label to identify the incoming/outgoing interface at each router for each tree, our method requires each *label-switched router* (LSR) to maintain a single label for all of these trees. This method allows one label to be required by the upstream router to identify the outgoing interface and another label to be required by the downstream router to identify the incoming interface only. As a result, smaller number of labels is used and smaller amount of entries is maintained at LSRs.

Furthermore, we will extend TSP to increase the degree of overlapping of trees in MPLS networks. Since the amount of reduction in the number of labels used and the amount of entries maintained at LSRs is dependent on how many tree branches are overlapping with each other, we can increase such reduction by making more and more branches to be overlapping with each other. To do so, we graft a set of disjoint branches to a selected branch and then prune the original

disjoint branches away. Using this tree switching technique, the set of disjoint branches is converted to a set of overlapping branches. Once these branches become overlapping with each other, the same label can be used to identify the interface. This allows further reduction in the number of labels used and the number of entries maintained at the routers.

Bibliography

E. Aharoni and R. Cohen, "Restricted Dynamic Steiner Trees for Scalable Multicast in Datagram Networks," in *Proc. of IEEE INFOCOM*, 1997.

F. Baccelli, D. Kofman, and J.L. Rougier, "Self Organizing Hierarchical Multicast Trees and Their Optimization," in *Proc. of IEEE INFOCOM*, pp. 1081-1089, 1999.

M. Baldi and Y. Ofek, "Ring versus Tree Embedding for Real-time Group Multicast," in *Proc. of IEEE INFOCOM*, pp. 1099-1106, 1999.

A. Ballardie, "Core Based Trees (CBT Version 2) Multicast Routing – Protocol Specification," *RFC 2189*, Sep. 1997.

A.J. Ballardie, P.F. Francis, and J. Crowcroft, "Core Based Trees (CBT): An Architecture for Scalable Multicast Routing," in *Proc. of ACM SIGCOMM*, San Francisco, 1993.

A. Banerjea, M. Faloutsos, and R. Pankaj, "Designing QoSMIC: A Quality of Service Sensitive Multicast Internet Protocol," *Internet Draft*, 1998.

S. Berson and S. Vincent, "Aggregation of Internet Integrated Services State," *Internet Draft (draft-berson-rsvp-aggregation-00.ps)*, 1998.

R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification," *RFC 2205*, 1997.

R. Briscoe and M. Tatham, "End to End Aggregation of Multicast Addresses," *Internet Draft (draft-briscoe-ama-00.txt)*, Nov. 1997.

B. Cain, S. Deering, and A. Thyagarajan, "Internet Group Management Protocol, Version 3," *Internet Draft (draft-ietf-idmr-igmp-v3-01.txt)*, Feb. 1999.

K. Calvert, E. Zegura, and M. Donahoo, "Core Selection Methods for Multicast Routing," in *Proc. of IC3N*, 1995.

J.O. Calvin, J. Seeger, G.D. Troxel, and D.J.V. Hook, "STOW Realtime Information Transfer and Networking System Architecture," in *Proc. of the 12th DIS Workshop*, 1995.

F.Y.Y. Cheng and R.K.C. Chang, "Design and analysis of scalable source-based Internet multicast routing protocols," in R.O. Onvural, S. Civanlar, P.J. Doolan, and J.V. Luciani (Eds.), *Internet Routing and Quality of Service: Proc. of SPIE*, vol. 3529, pp. 319-328, Dec. 1998.

F.Y.Y. Cheng and R.K.C. Chang, "A Tree Switching Protocol for Multicast State Reduction," to appear in *Proc. of the 5th IEEE Symposium on Computers and Communications*, Antibes, France, July 2000.

S. Deering, "Host Extensions for IP Multicasting," *RFC 1112*, 1989.

S. Deering, *Multicast Routing in a Datagram Internetwork*, Ph.D. Thesis, Stanford University, 1991.

S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei, "The PIM Architecture for Wide-area Multicast Routing," *IEEE/ACM Trans. on Networking*, vol. 4, no. 2, 1996.

M.J. Donahoo, K.L. Calvert, and E.W. Zegura, "Center Selection and Migration for Wide-Area Multicast Routing," *Journal of High Speed Networks*, vol. 6, pp. 141-164, 1997.

S. Engevall, M. Gothe-Lundgren, and P. Varbrand, "A Strong Lower Bound for the Node Weighted Steiner Tree Problem," *Networks*, vol. 31, pp. 11-17, 1998.

H. Eriksson, "MBone: The Multicast Backbone," *ACM Communications*, vol. 37, no. 8, pp. 54-60, 1994.

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

D. Estrin, D. Farinacci, V. Jacobson, C. Liu, L. Wei, P. Sharma, and A. Helmy, "Protocol Independent Multicast – Dense Mode (PIM-DM): Protocol Specification," *Internet Draft*, 1995.

D. Estrin, M. Handley, A. Helmy, P. Huang, and D. Thaler, "A Dynamic Bootstrap Mechanism for Rendezvous-based Multicast Routing," in *Proc. of IEEE INFOCOM*, pp. 1090-1098, 1999.

M. Faloutsos, A. Banerjea, and R. Pankaj, "QoSMIC Quality of Service sensitive Multicast Internet ProtoCol," in *Proc. of ACM SIGCOMM*, pp. 144-154, 1998.

D. Farinacci, Y. Rekhter, P. Lothberg, H. Kilmer, and J. Hall, "Multicast Source Discovery Protocol (MSDP)," *Internet Draft (draft-farinacci-msdp-00.txt)*, June 1998.

S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing," *IEEE/ACM Transactions on Networking*, 1996.

J.J. Garcia-Luna-Aceves and E.L. Madruga, "A Multicast Routing Protocol for Ad-Hoc Networks," in *Proc. of IEEE INFOCOM*, 1999.

E. Gelenbe, A. Ghanwani, and V. Srinivasan, "Improved Neural Heuristics for Multicast Routing", *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 2, pp. 147-155, 1997.

M.X. Goemans and Y. Myung, "A Catalog of Steiner Tree Formulations," *Networks*, vol. 23, pp. 19-23, 1993.

L. Guo and I. Matta, "On State Aggregation for Scalable QoS Routing," in *Proc. of IEEE ATM'98 Workshop*, 1998.

M. Handley, "Session Directories and Scalable Internet Multicast Address Allocation," in *Proc. of ACM SIGCOMM*, pp. 105-117, 1998.

M. Handley, J. Crowcroft, and I. Wakeman, "Hierarchical Protocol Independent Multicast," *ftp://cs.ucl.ac.uk/darpa/hpim.ps.gz*, 1997.

A. Helmy and D. Estrin, "'STRESS' Testing Applied to a Multicast Routing Protocol," in *Proc. of IEEE INFOCOM*, 1998.

H.W. Holbrook and D.R. Cheriton, "IP Multicast Channels: EXPRESS Support for Large-scale Single-source Applications," in *Proc. of ACM SIGCOMM*, 1999.

S.-P. Hong, H. Lee, and B.H. Park, "An Efficient Multicast Routing Algorithm for Delay-Sensitive Applications with Dynamic Membership," in *Proc. of IEEE INFOCOM*, pp. 1433-1440, 1998.

C. Huitema, *Routing in the Internet*, Prentice Hall, Inc., 1995.

X. Jia, "A Distributed Algorithm of Delay Bounded Multicast Routing for Multimedia Applications in Wide Area Networks," *IEEE/ACM Trans on Networking*, vol.6, no.6, Dec. 1998, pp. 828-837.

X. Jia, J. Cao, and W. Jia, "A Classification of Multicast Mechanisms: Implementations and Applications," *The Journal of Systems and Software*, vol. 45, no. 2, Mar. 1999, pp. 99-112.

X. Jia, N. Pissinou, and K. Makki, "A Real-Time Multicast Routing Algorithm for Multimedia Applications," *Computer Communications Journal*, vol. 20, no. 12, Nov. 1997, pp.1098-1106.

X. Jia and L.Wang, "A Group Multicast Routing Algorithm by Using Minimum Multiple Steiner Trees," *Computer Communications Journal*, vol. 20, no. 9, Sep. 1997, pp.750-758.

X. Jia, Y. Zhang, and K. Makki, "A Distributed Multicast Routing Protocol for Real-Time Multicast Applications," *Computer Networks and ISBN Systems*, vol. 31, no. 1-2, Jan. 1999, pp. 113-121.

R.M. Karp, "Reducibility among Combinatorial Problems," in R.E. Miller and J.W. Thatcher (Eds.), *Complexity of Computations*, New York: Plenum Press, pp. 85-104, 1972.

V.P. Kompella, J.C. Pasquale, and G.C. Polyzos, "Multicast Routing for Multimedia Communication," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 286-292, June 1993.

L. Kou, G. Markowsky, and L. Berman, "A Fast Algorithm for Steiner Trees," *Acta Informatica*, vol. 15, pp. 141-145, 1981.

S. Kumar, P. Radoslavov, D. Thaler, C. Alaettinoglu, D. Estrin, and M. Handley, "The MASC/BGMP Architecture for Inter-domain Multicast Routing," in *Proc. of ACM SIGCOMM*, pp. 93-104, 1998.

J. Liebeherr and B.S. Sethi, "A Scalable Control Topology for Multicast Communications," in *Proc. of IEEE INFOCOM*, pp. 1197-1204, 1998.

H.-C. Lin and S.-C. Lai, "VTDM – A Dynamic Multicast Routing Algorithm," in *Proc. of IEEE INFOCOM*, pp. 1426-1432, 1998.

D.H. Lorenz and A. Orda, "QoS Routing in Networks with Uncertain Parameters," in *Proc. of IEEE INFOCOM*, pp. 3-10, 1998.

J. Moy, "Multicast Extensions to OSPF," *RFC 1584*, Mar. 1994.

J. Moy, "OSPF Version 2," *RFC 2328*, Apr. 1998.

J.-J. Pansiot and D. Grad, "On Routes and Multicast Trees in the Internet," in *SIGCOMM Computer Communication Review*, vol. 28, no. 1, Jan. 1998.

M. Parsa and J.J. Garcia-Luna-Aceves, "A Protocol for Scalable Loop-Free Multicast Routing," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 3, pp. 316-331, Apr. 1997.

M. Parsa and J.J. Garcia-Luna-Aceves, "Scalable Internet Multicast Routing," in *Proc. of ICCN*, pp. 162-166, 1995.

R. Perlman, C.-Y. Lee, A. Ballardie, J. Crowcroft, Z. Wang, and T. Maufer, "Simple Multicast: A Design for Simple, Low-Overhead Multicast," *Internet Draft (draft-perlman-simple-multicast-02.txt)*, 1999.

J.M. Pullen and E.L. White, "Dual-Mode Multicast for DIS," in *Proc. of the 12th DIS Workshop*, 1995.

P. I. Radoslavov, D. Estrin, and R. Govindan, "Exploiting the Bandwidth-Memory Tradeoff in Multicast State Aggregation", *Tech. Rep. 99-697, USC Computer Science Department*, 1999.

S. Ramanathan, "Multicast Tree Generation in Networks with Asymmetric Links," *IEEE/ACM Trans. on Networking*, vol. 4, no. 4, pp. 558-568, 1996.

K. Ravindran and T.-J. Gong, "Cost Analysis of Multicast Transport Architectures in Multiservice Networks," *IEEE/ACM Trans. on Networking*, vol. 6, no. 1, pp. 94-109, Feb. 1998.

G.N. Rouskas and I. Baldine, "Multicast Routing with End-to-End Delay and Delay Variation Constraints," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 3, pp. 346-356, 1997.

H. Salama, *Multicast Routing for Real-Time Communication on High-Speed Networks*, Ph.D. Thesis, Dept. of Electrical and Computer Engineering, North Carolina State University, 1996.

H.F. Salama, D.S. Reeves, and Y. Viniotis, "A Distributed Algorithm for Delay-Constrained Unicast Routing," in *Proc. of IEEE INFOCOM*, 1997.

C. Shields, *Ordered Core Based Tree*, MSc. Dissertation, University of California, Santa Cruz, June 1996.

C. Shields and J.J. Garcia-Luna-Aceves, "The Ordered Core Based Tree Protocol," in *Proc. of IEEE INFOCOM*, Kobe, Japan, 1997.

C. Shields and J.J. Garcia-Luna-Aceves, "The HIP Protocol for Hierarchical Multicast Routing," in *Proc. of Seventeenth Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC'98)*, Puerto Vallarta, Mexico, July 1998

R. Sriram, G. Manimaran, and C.S.R. Murthy, "A Rearrangeable Algorithm for the Construction of Delay-Constrained Dynamic Multicast Trees," in *Proc. of IEEE INFOCOM*, pp. 1073-1080, 1999.

W.R. Stevens, *TCP/IP Illustrated*, Addison-Wesley, 1994.

D. Thaler, D. Estrin, and D. Meyer, "Border Gateway Multicast Protocol (BGMP): Protocol Specification," *Internet-Draft: draft-ietf-idmr-gum-02.txt*, 1998.

D. Thaler and M. Handley, "On the Aggregation of Multicast Forwarding State," to appear in *Proc. of IEEE INFOCOM*, 2000.

D.G. Thaler and C.V. Ravishankar, "Distributed Center-Location Algorithms: Proposals and Comparisons," in *Proc. of IEEE INFOCOM*, pp. 75-84, 1996.

A. Thyagarajan and S. Deering, "Hierarchical Distance-Vector Multicast Routing for the Mbone," in *Proc. of ACM SIGCOMM*, 1995.

J. Tian and G. Neufeld, "Forwarding State Reduction for Sparse Model Multicast Communication," in *Proc. of IEEE INFOCOM*, pp. 711-719, 1998.

H. Tode, Y. Sakai, M. Yamamoto, H. Okada, and Y. Tezuka, "Multicast Routing Algorithm for Nodal Load Balancing," in *Proc. of IEEE INFOCOM*, pp. 2086-2095, 1992.

P. Tsuchiya, "Efficient and Flexible Hierarchical Address Assignment," *TM-ARH-018495*, Bellcore, February 1991.

D. Waitzman, C. Partridge, and S. Deering, "Distance Vector Multicast Routing Protocol," *RFC 1075*, 1988.

B.M. Waxman, "Routing of Multipoint Connections," *IEEE J. Sel. Areas in Communications*, vol. 6, no. 9, pp. 1617-1622, 1988.

B.M. Waxman, "Performance Evaluation of Multipoint Routing Algorithms," in *Proc. of IEEE INFOCOM*, pp. 980-986, 1993.

L. Wei and D. Estrin, "The Trade-offs of Multicast Trees and Algorithms," in *Proc. of ICCCN*, 1994.

L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A New Resource ReSerVation Protocol," *IEEE Network*, 1993.

Q. Zhu, M. Parsa, and J.J. Garcia-Luna-Aceves, "A Source-Based Algorithm for Delay-Constrained Minimum-Cost Multicasting," in *Proc. of IEEE INFOCOM*, 1995.