QoS-Aware Hierarchical Multicast Routing on Next Generation Internetworks

by

Satyabrata Pradhan

A thesis

Submitted to the Faculty of Graduate Studies
in Partial Fulfillment of the Requirements
for the degree of

MASTER OF SCIENCE

Department of Computer Science
University of Manitoba
Winnipeg, Manitoba, Canada

© Satyabrata Pradhan, 2000

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisitions et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-51787-X

Canada

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION PAGE

QoS-Aware Hierarchical Multicast Routing on Next Generation Internetworks

BY

Satyabrata Pradhan

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirements of the degree of

Master of Science

SATYABRATA PRADHAN ©2000

Permission has been granted to the Library of The University of Manitoba to lend or sell copies of this thesis/practicum, to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film, and to Dissertations Abstracts International to publish an abstract of this thesis/practicum.

The author reserves other publication rights, and neither this thesis/practicum nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

Abstract

Several emerging applications of the current and next generation Internet, e.g. IP radio, IP TV, and video and audio conferencing, etc., involve one-to-many and many-to-many data communications called multicasting. Many real-time applications over IP, such as IP Telephony and video conferencing, can not operate with the best effort service provided by current IP networks. To support such applications, IP networks need to provide certain quality of service (QoS) assurances. Most of QoS-based multicast routing protocols are based on a "flat" routing model that do not scale well for large size networks.

This thesis proposes QHMRP, a novel QoS-aware hierarchical multicast routing protocol. The scalability issue is addressed by organizing the network as a hierarchy of domains using the full-mesh aggregation technique. The concept of domain *controller* is used for coordinating the creation and maintenance of multicast trees. The protocol uses a novel reverse flooding approach to connect host routers with the tree while satisfying end-to-end QoS constraints. This is a distributed algorithm, which uses only local state information at each router. The worst-case connection time and message overhead are estimated and analysis shows that QHMRP constructs loop-free multicast trees. Simulation results show that the message overhead of QHMRP is much smaller than that of the flat routing protocol using reverse flooding.

Acknowledgements

I express my gratefulness to Prof. M. Maheswaran for his valuable guidance throughout the research project. I am also thankful to the thesis committee members, Prof. P. Graham and Dr. J. Rueda, for their useful suggestions. Finally, I express my hearty gratitude to my wife (Nishi) and son (Sougat) who have been the source of much needed motivation and support during the many months of work this thesis has entailed.

Table of Contents

List of Figures

List of Tables

1 INTRODUCTION

Routing is a key important operation for successful data transmission in packet switching networks e.g. IP networks [Sta98]. Routing algorithms can be grouped into two fundamental types: unicast routing and multicast routing. In unicast routing, packets are transmitted from a single source to a single destination. Multicast routing transmits packets from one or multiple sources to multiple receivers that have been configured as members of a multicast group in various scattered subnetworks. Examples of multicast include the transmission of corporate messages to employees, communication of stock quotes to brokers, video and audio conferencing, and replicating databases and web site information, etc. IP multicast supports this type of transmission by enabling sources to send a single copy of a message to multiple recipients who explicitly want to receive the information. This is far more efficient than requiring the source to send an individual copy of a message to each requester (referred to as point-to-point unicast).

IP networks provide best effort service that is subject to unpredictable delay and potential data loss. Many real-time applications over IP, such as IP Telephony, radio and television over IP, video conferencing etc., can not operate with the best effort service provided by current IP networks. To service these applications, IP networks need to provide some *quality of service* (QoS) guarantees. QoS is defined as the ability of network elements (e.g. an application, host, and router) to provide some level of assurance that the data

traffic can meet certain service requirements (e.g. delivery time). Based on application requirements, QoS can be divided into two basic types: resource reservation and prioritization. To accommodate the need for these different types of QoS, a number of QoS protocols and algorithms have been developed or are under development including *resource reservation* (RSVP) [ZhD93], *differentiated services* (DiffServ), and *integrated services* (IntServ) [Sta99]. QoS aware protocols such as the one developed in this thesis can be implemented using new IP switching schemes such as *multi protocol labeling switching* (MPLS) [Sta99].

The importance of QoS-based multicast routing has prompted several research initiatives in this area. Most of these protocols are based on "flat" routing schemes [ChN98a, FaB98], which model the entire network as a single domain. These protocols do not scale well for large size networks. The scalability issue can be addressed by organizing the network in to a hierarchy of domains [ShG98, MoV98]. In this approach, the network is structured as L levels of domains (called an L-level hierarchy). Each level in the hierarchy consists of multiple domains. A domain in level-i is called an i-domain and routers form 0-domain. A number of routers (i.e. 0-domains) are grouped together to form a 1-domain and, in general, a group of domains from level-i are grouped to form an (i+1)-domain.

This research focuses on QoS-based hierarchical multicast routing for large IP networks, a problem that has not been addressed to date. The proposed protocol, called *QoS-based Hierarchical Multicast Routing Protocol* (QHMRP), uses the "full-mesh" approach

[Lee95] to organize the network into multiple levels where a domain is represented by its border routers in a higher level domain. The concept of domain *controller* is used for coordinating the creation and maintenance of shared multicast trees. The protocol proposes a novel *reverse flooding* approach to connect new hosts with the tree while satisfying end-to-end QoS constraints. The concepts presented in this thesis for shared multicast tree can also be used for constructing source-based trees. A brief outline of the protocol for source-based tree is also presented.

The thesis is organized as follows. The next chapter reviews the related work done in the area of multicasting, QoS, and hierarchical routing. The scope of the proposed research is also described in Chapter 2. This is followed by the detailed description of the network model in Chapter 3. The modeling includes the definition of the hierarchy, naming convention for the routers, aggregation techniques, and description of the QoS state metrics used in the study. Chapter 4 presents a detailed description and analysis of the proposed protocol. This is followed by simulation results in Chapter 5. Finally, Chapter 6 summarizes the thesis and presents directions for future work.

2 RELATED WORK

2.1 Preliminary Remarks

The Internet routing being a major research area, a considerable amount of literature has been developed on routing protocols, which has been reviewed quite effectively in [DiD97, ChN98a]. However, since QoS-based multicasting is a new area, there are very few publications on this topic. In fact, there are no publications in the open literature on QoS-based hierarchical multicast routing. The objective in this section is to briefly touch upon the more important contributions that are directly relevant to the thesis. Therefore, this literature review includes non QoS-based multicasting and hierarchical routing as well as QoS-based hierarchical unicasting.

2.2 Multicast Routing

Multicast routing protocols construct a routing tree connecting all the senders and receivers of the multicast group. There are two basic types of multicast trees: source-based trees and shared trees. In the source-based tree approach, the protocol computes an implicit spanning tree for each source in the multicast group. In the shared tree approach, a single spanning tree is shared by all the group members to send and/or receive messages. Having only one delivery tree for multiple sources may result in non-optimal routes and cause delays in message delivery. The primary advantage of the shared tree

approach is that it conserves network resources and, therefore, offers more favorable scaling characteristics than the source based approach.

There are several basic algorithms that may be employed by multicast routing protocols to construct and maintain multicast trees [DiD97, ChN98a]. This section briefly discusses these algorithms as well as the prevalent multicast routing protocols.

2.2.1 Multicast Routing Algorithms

Multicast routing requires some distribution tree rather than a simple point-to-point path through the network. The objective of multicast routing algorithms is to construct and maintain the distribution tree, called the multicast tree. The routing algorithms can broadly be classified as source routing and distributed routing. In the source routing, each router maintains the complete global state of the network. Based on the global state, the multicast tree is locally computed at the source router. In distributed routing, the tree is computed by an algorithm distributed over different routers in the network.

The simplest technique for multicasting is flooding. In this approach when a packet arrives at a router for the first time, the router forwards the packet on all interfaces except the one on which it arrived. Otherwise, the router simply discards the packet. A flooding algorithm is very simple to implement. However, flooding does not scale for large networks and makes inefficient use of router memory since each router is required to maintain a distinct table entry for each recently seen packet. The flooding approach has been applied for link state information advertisement in the Open Shortest Path First

(OSPF) protocol [Moy98]. An improved version of flooding has also been used to develop a distributed algorithm for QoS based routing [ChN98].

The spanning tree algorithm is a refinement of simple flooding to provide a more efficient routing. A spanning tree is a subset of the Internet that spans all nodes in the internetwork. The spanning tree can also be generated by Dijkstra's algorithm [Sta98]. A spanning tree solution is relatively easy to implement. However, it has two drawbacks: it centralizes all the traffic on a small set of links and it does not consider the multicast group membership. The Reverse Path Forwarding (RPF) algorithm builds a group-specific spanning tree for each potential source subnetwork [DaM78]. Pruning techniques have been proposed to generate a per-source-group multicast tree from the per-source broadcast tree (i.e. spanning tree) generated by RPF [DeC90]. This "broadcast and prune" technique, which is used in DVMRP, is a distributed algorithm and is called Reverse Path Multicasting (RPM) [Pus99].

The Steiner algorithm designs a tree that spans the multicast group members and minimizes a cost function defined on the network edges. It is a centralized algorithm. A number of heuristics have been proposed for distributed calculation of the Steiner tree [Win87, KoM81, HwR92]. Although Steiner trees minimize network resources for constructing a delivery tree, difficulties in computation has made these trees of little practical importance.

Most of the previous algorithms for constructing multicast tree generate a source-rooted tree for each (source, group) pair. These approaches are suitable for single sender/fixed recipient scenarios. However, for multiple senders/multiple recipient cases, it is more appropriate to use a single shared tree that can be used by all group members to send and receive the multicast packets. The Core Based Tree (CBT) algorithm is an example of this approach [Bal97, Bal97a]. A single router or a set of routers is chosen to be the *core* router of the delivery tree. When a host wants to receive messages from and/or send messages to a multicast tree it joins the core of the multicast group. Since CBT constructs only one delivery tree for each multicast group, routers are required to keep less information as compared with other algorithms. CBT also conserves network bandwidth by forwarding packets only along the shared tree (it does not use flooding). However, using a single tree for each group may lead to traffic concentration and bottlenecks around the core router. Selection and management (in case of core failure) of the core router are additional problems. This approach has also been used in the PIM-SM protocol [EsF98].

2.2.2 Multicast Routing Protocols

A multicast routing protocol uses one or more routing algorithms to construct and maintain the multicast tree. Some of the most widely used protocols are reviewed in this section. The Distance Vector Multicast Routing Protocol (DVMRP) [Pus99] is a distributed algorithm that dynamically generates a multicast delivery tree for each (*source, group*) pair using the RPM technique [DeC90]. It is an extension of the Routing Information Protocol (RIP) [Hed88] and can be summarized as a "broadcast and prune"

multicast routing protocol. In this approach, the first datagram for any (source, group) pair is flooded across the entire Internet to create a spanning tree. The initial datagram is delivered to all leaf routers, which transmit prune messages back towards the source if there is no multicast group member on their directly attached leaf subnetwork. The prune messages remove all branches from the tree that do not lead to group members, thus creating a source-rooted shortest path (based on distance vector) tree with all leaves having group members. After a period of time, the pruned branches grow back (called *grafting*), and the next datagram for the (source, group) pair is forwarded across the entire Internet, resulting in a new set of prune messages. This method takes care of changes in the multicast group membership over time. DVMRP supports tunnel interfaces and is currently deployed in the majority of MBone routers.

As discussed earlier, the CBT protocol [Bal97, Bal97a] builds a shared multicast distribution tree per multicast group. As all the routers connect to the core, the protocol may lead to traffic concentration and a performance bottleneck around the core router. Several core management approaches, e.g. core selection, core failure handling, and core migration [FlH98, HuF98], have been proposed to avoid these problems. Another solution to avoid the performance bottleneck is to use multiple cores. Unfortunately, when multi-core architecture is used, the CBT protocol can form loops and thus fail to build a connected multicast tree, even when the underlying routing is stable. The Ordered Core Based Tree (OCBT) protocol [ShG97] eliminates these deficiencies. The performance bottleneck around the core router can also be eliminated by allowing the new members to join any one of the on-tree routers instead of joining the core [FaB98].

Unlike other protocols, the Protocol Independent Multicast (PIM) routing protocol does not require the existence of any specific underlying unicast protocol. It can work with any unicast protocol. PIM contains two protocols: PIM-Dense Mode (PIM-DM) [EsF96], which is more efficient when the group members are densely distributed, and PIM-Sparse Mode (PIM-SM) [EsF98], which performs better in cases where group members are sparsely distributed. The operations of PIM-DM and PIM-SM are integrated so that a single router can run different modes for different multicast groups.

PIM-DM is similar to DVMRP in that it employs the RPM algorithm. But unlike DVMRP, which uses the routing table to calculate the distance vector to flood the datagram at each node, PIM-DM simply forwards multicast traffic on all downstream interfaces until explicit prune messages are received. This way, PIM-DM eliminates routing protocol dependencies at the cost of packet duplication. Like DVMRP, PIM-DM uses graft message to add new members to the group.

PIM-SM is a receiver initiated protocol where a host router joins a multicast group by sending explicit join message to a Rendezvous Point (RP), which is similar to the concept of the core in CBT. Like CBT, PIM-SM does not generate an optimal shared tree. The receiver may also decide to switch from the shared tree to a source-based shortest path tree. Even in this case, the source continues to send its data to the RP for other possible receivers. The ability to switch from an RP-rooted shared tree to a source-based tree is the main difference between PIM-SM and CBT.

2.3 Hierarchical Routing

The routing protocols discussed so far organize the entire Internet into a single domain and do not scale well with increasing network size. Several hierarchical protocols have been proposed to address the scalability problem associated with flat routing schemes. Hierarchical protocols that use link state information and routing tables use topology aggregation for increasing scalability. Topology aggregation is achieved by grouping neighboring network nodes into routing domains and representing the routing information for each domain in an aggregated (and therefore, compact) manner. The aggregated information is used by network nodes outside the domain to make routing decisions. Different techniques such as symmetric star, full-mesh, spanning tree, and complex node representation have been used for topology aggregation [Lee95].

Hierarchical DVMRP [ThD95] and Multicast extensions for OSPF (MOSPF) [Moy94, Moy94a] are currently being used in the Internet for hierarchical multicasting. Hierarchical DVMRP divides the Internet into a number of domains, thus, creating a two-level hierarchy. The intra-domain multicasting may run any protocol while the inter-domain multicasting runs DVMRP for routing between the *border routers* (BR) of different domains. MOSPF organizes the internet into a three-level hierarchy: the Internet is divided into *autonomous systems* (AS) and each AS is further divided into subgroups called *areas*. Dijkstra's algorithm is used to construct shortest path delivery trees routed at the source nodes for intra-area routing. The inter-area multicast forwarders

and inter-AS multicast forwarders are used for inter-area and inter-AS multicasting, respectively [Moy94].

Unlike the above hierarchical protocols, the *area-based link-vector algorithm* (ALVA) [BeG98] organizes the Internet into an *N*-level hierarchy. Each router maintains a topology table, which contains complete link vector information for its own domain and aggregated information for all the neighboring domains. The symmetric star approach [Lee95] is used for aggregating the link vector information, which is distributed over the network by flooding. ALVA uses Dijkstra's algorithm to compute the path for unicast routing based on the topology table stored at each router.

The *hierarchical PIM* (HPIM) protocol [HaC] is based on PIM-SM and constructs multicast shared trees for *N*-level hierarchical networks. Unlike PIM-SM, HPIM does not require advertisement of the *rendezvous point* (RP). Each level in the hierarchy has one candidate RP (along with backup RPs in case of failure of the primary) for each multicast group. Every router knows the address of the candidate RPs in its level and each candidate RP knows the address of the candidate RPs in the level above it. When a host wants to join the multicast tree, it joins the candidate RP in its level. Each RP in sequence joins to the RP in the level above it until an on-tree RP (i.e. RP that is part of the multicast tree) is reached.

The HIP protocol uses *ordered core based trees* (OCBT) for interdomain multicast routing in hierarchical networks [ShG98]. It uses the concept of a virtual *center point*

(CP) which is defined as a domain containing the actual CP. There is one virtual CP in each level of the hierarchy. The key differences between HPIM and HIP are in the process of disseminating the CP location and finding the CP during the joining process. Both HPIM and HIP provide interdomain protocols which are capable of operating with existing intradomain protocols like DVMRP or MOSPF.

2.4 QoS-based Routing

As discussed earlier, the convergence of different real-time applications over IP demands certain QoS guarantees. Different routing algorithms have been proposed to meet the QoS requirements of different applications [ChN98a]. Selective probing is a distributed QoS-based unicast routing protocol that selectively floods along links, which can meet the QoS requirements, to find a path between a source and a destination. The number of flooding messages is controlled by allowing only one flooding message belonging to a particular (*source*, *destination*) pair to pass through a router. However, this approach may fail to find a tentative path even when such a path exists [ChN98]. This problem is taken care of by introducing a delay, which is same as the node delay, at each router while forwarding the messages. The path followed by the first message that arrives at the destination is then selected as the routing path.

The QoS-aware multicast routing protocol (QMRP) [ChN99] constructs a shared tree by unicasting *Request* message from the host router towards the core. If a router in the unicast path does not satisfy the QoS requirement, the *Request* message back tracks one router and is then sent along all other available paths as unicast messages towards the

core. When an on-tree router (i.e. a router that is part of the multicast tree) or the core receives a *Request* message it sends an *Ack* message back to the host router. After receiving all *Ack* messages, the host router selects a path to connect to the tree.

The QMRP may not be able to find a feasible path for additive (e.g. delay) QoS requirements, even if such a path exists. Consider a scenario where there is only one feasible path that satisfies the delay requirement. While unicasting, the *Request* message deviates from the feasible path and does not satisfy the QoS requirement when it is more than one hop away from the feasible path, then the message may never come back to the feasible path. Therefore, the *Request* message may never reach the core or any other on-tree router.

The QoS sensitive Multicast Internet protoCol (QoSMIC) [FaB98] uses the concept of manager to construct a shared multicast tree. It uses two approaches to join a multicast tree: local search and multicast tree search. In local search, flooding is used to join an on-tree router. In multicast tree search, the host router sends an *M-Join* message to the manager, which knows the addresses of all on-tree routers. The manager selects certain on-tree routers and asks them to unicast *Bid* messages towards the host router. After receiving the *Bid* messages, the host router selects the best path to connect to the multicast tree.

Unlike CBT or PIM-SM, where new hosts are connected to a single router (CP or RP), the QoSMIC protocol connects new hosts to one of the on-tree routers. Therefore, it

avoids the performance bottleneck around the "core" router. However, like QMRP, QoSMIC uses unicasting of *Bid* messages to find a feasible path and, therefore, may not find an existing feasible path that satisfies QoS requirement.

The protocols in [MoV98] and [GuO97] address the unicast "loose" source routing problem with bandwidth and end-to-end delay constraints in hierarchical networks. In loose source routing, only the high level path is specified by the source. The detailed path through a remote subnetwork is determined by a border router of that network. The protocol by Montgomery and Veciana [MoV98] uses a full-mesh approach while Guerin and Orda [GuO97] use the symmetric star approach for aggregation.

The viewserver hierarchy is presented in [AlS95] for QoS-based interdomain unicast routing. In this scheme, the domain level views of the domain and some of the neighbouring domains are kept only at certain special nodes called *viewservers*. A view server provides domain level source routing between source and destination nodes in a single view (the domains covered by the viewserver). Obtaining source route between routers not in a single view involves accumulating the views of a sequence of viewservers. The protocol handles topology changes such as node/link failure, link cost change, and domain partition. Border routers detect these changes and communicate to viewservers by flooding.

2.5 Scope of the Research

As the size of the Internet continues to grow, the scalability of flat routing protocols becomes a major limitation. Hierarchical routing provides a solution to this problem. Furthermore, QoS based multicast routing is essential to the success of different real-time applications over IP involving multiple senders and receivers. For multicast groups with large numbers of senders and receivers, such as video and audio conferencing, the shared tree approach can save significant network resources compared to the source-based tree approach. This thesis addresses these issues and develops a new QoS-based multicast routing protocol for hierarchical networks. The proposed protocol, called the *QoS-aware Hierarchical Multicast Routing Protocol* (QHMRP), can construct both source-based and shared multicast trees with end-to-end QoS guarantees. This is the first protocol, which deals with QoS-aware hierarchical routing. The thesis presents the detailed protocol and simulation results for the QHMRP for shared multicast trees and an outline of the QHMRP for source-based multicast trees.

In the proposed approach, the routers in the network are organized into an N-level hierarchy. Each level in the hierarchy consists of a number of domains. Each domain has at least one border router that defines the edge of the domain and is also connected to other external domains. One of the border routers is selected as the *controller* for the domain. The controllers store information about the multicast trees within its domain and coordinate the joining process. Each router in the network belongs to only one domain in a level and keeps topology information of the domain to which it belongs. The full-mesh

approach is used to aggregate the network topology where a domain is represented by its border routers in a higher level domain. If a domain contains any on-tree router, then its controller stores the topology of the corresponding tree branch within the domain. All the controllers of higher level domains store the address of their subdomain controller(s) that contains the multicast tree. This approach guarantees that at least one of the controllers in each level of the hierarchy knows the location of the multicast tree, if it exists.

When a host wants to join a shared multicast tree, it sends a *JoinRequest* to its controller. If the controller is aware of the multicast tree, it initiates the join process, otherwise the *JoinRequest* is forwarded to a higher level controller, and so on. When a controller that is aware of the multicast tree receives the *JoinRequest* message, it requests all on-tree routers in its domain to flood *Flooding* messages towards the host router. The flooding is controlled at each router by forwarding the join messages only along those links, which satisfy three forwarding conditions. The "hierarchical" forwarding condition guarantees that the flooding is limited to the lowest level domain that contains the host router and on-tree routers. The "topological" and "QoS" forwarding conditions flood along links which lead to the host router, and satisfy the QoS requirement, respectively.

The flooding messages need to be further controlled to guarantee that the protocol finds a feasible path, if it exists. This can be done by introducing a delay at each router while forwarding a packet [ChN98]. QHMRP proposes two additional techniques to control the flooding. In the first alternate approach, multiple messages belonging to the same (*source, destination*) pair are allowed to pass through a router if they satisfy certain QoS

constraints. The second approach uses a combination of the above two methods. The path followed by the *Flooding* message arriving first at the host router is selected as the feasible path for connection with the tree. This method of flooding from the on-tree routers towards the host router is referred to as *reverse flooding*.

QHMRP considers both bandwidth and delay based end-to-end QoS requirements. Analysis is done to estimate the worst-case connection time and message overhead and to show that the proposed protocol is loop-free. The feasibility and performance of the proposed protocol is assessed by simulation. The advantages of QHMRP over the flat routing scheme using reverse flooding is also studied by simulation.

QHMRP is the first QoS-aware protocol for multicast routing in hierarchical networks. In addition, this thesis proposes new forwarding conditions and techniques to control flooding. These features can be used to improve any protocol using a flooding algorithm for routing.

3 SYSTEM MODEL

3.1 Preliminary Remarks

Organization of the Internet and the naming scheme used to uniquely identify every router are the key features in any hierarchical routing protocol. Furthermore, every attempt should also be made to reduce the message overhead of connecting to the multicast tree. As will be seen in the next chapter, QHMRP can use topological information stored in routing tables to reduce message overhead while finding feasible paths between a host router and a multicast tree. Topology aggregation is the most important technique for achieving scalability of protocols using routing. This chapter presents the hierarchical organization and the naming scheme used to model the Internet. It also presents the aggregation technique and QoS state metrics used in the proposed protocol.

3.2 Internetwork Model

The internetwork is modeled as a directed graph in which the routers are the nodes and direct links between the nodes are the edges of the graph. Each router has input and output queues, and the capability to process messages. The processing and queueing time at each node is modeled by a delay parameter for each link connected to the node. Each edge in the graph has two costs, one in each direction, associated with it. In the hierarchical model, the routers in the network are organized into L levels of domains. A

domain in level i is called an i-domain and routers form 0-domains. A group of routers (i.e. 0-domains) are grouped together to form a 1-domain and a group of i-domains are grouped together to form a $(i+1)$-domain. Each level in the hierarchy consists of multiple domains. Each router belongs to only one domain in a level but may belong to multiple levels.

The nodes in a domain are called the children of the domain and the domain is called the parent domain of the routers. The routers that define the edge of a domain and are also connected to other external domains are called *border routers* (BRs). For each i-domain, $i > 0$, one border router is selected to be the *controller* for a multicast group for the domain. This selection can be dynamic using some election process or it can be pre-selected. The controller of the parent domain of a router is called as the *parent controller*. It is assumed that the address of the parent controller is either specified to the routers or can be obtained by inquiring the "Session Directory" [HaJ96]. The controller keeps track of the addresses of the controllers of its sub-domains and the addresses of all the on-tree routers within its domain. The controller only helps in locating the on-tree routers and does not participate in the multicast tree. Therefore, the controller will not be a performance bottleneck like the core is in CBT [Bal97] and the RP is in PIM-SM protocols [EsF98].

An n-tuple addressing scheme is used to uniquely identify a router in the network. The address of a router is expressed as $(i_{L-1}.i_{L-2}. \ldots i_3.i_2.i_1.i_0)$, where i_j, $j = 0,1,2, \ldots, (L-2), (L-1)$, are nonzero positive integers. Here, i_j is the number of the sub-domain of a (j+1)-

domain to which the router belongs. An example of a 3-level hierarchical network showing the router numbering scheme is presented in Figure 1.

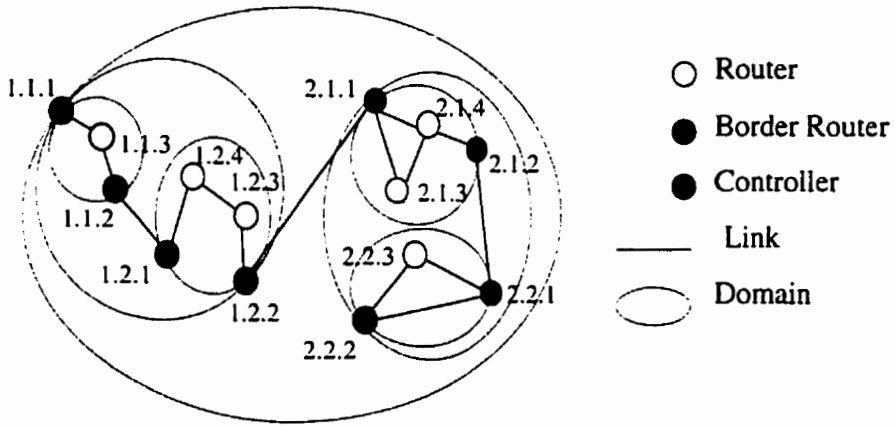

Figure 1 Hierarchical network model.

3.3 Topology Aggregation

The QHMRP needs to send unicast messages between routers and their parent controller. It is assumed that the network provides an underlying unicast protocol. In addition, QHMRP also uses reverse flooding (Section 4.3) to find feasible paths between a host router and a multicast tree. The message overhead during reverse flooding is reduced by flooding along links, which lead to the host router (Section 4.3.2). This requires routing tables at each router to store information on the domain topology. The routing table at a router contains the minimum distance between the router and all other routers in the domain via different neighboring routers.

The number of routers within a domain and, hence. the size of the routing table at each router increases with the level of the domain in the hierarchy. For example, the top-most

(level-3) domain in Figure 1 contains all the routers in the figure, while one of the level-1 domains contains only a few routers. Topology aggregation is used to achieve scalability by reducing the size of the routing tables. In this approach, each router stores multiple routing tables, one for each level it belongs to. For example, a router belonging to an i-domain has i routing tables for levels 1, 2, ..., i. The routing table for level k is referred to as $[R_k]$ and contains information on all routers belonging to the same k-domain as the storing router. The routing table is defined as:

$$[R_k] = [R_k(i, j)] \in \Re^{(N-1)\times d}$$

where $R_k(i,j)$ is the minimum distance (i.e. hop count) between the current router and the i^{th} k-domain router via the j^{th} k-domain neighbor. N is the number of routers in the k-domain and d is the k-domain degree of the current router (i.e. degree of the current router in the aggregated topology of the level-k domain).

There are two general ways to build a hierarchical network: a tree of trees; and trees within trees. In the first approach, the leaf nodes of a higher-level tree can each be the root node of a lower level tree. Each node can store the topology of the entire network where a foreign domain or a sub-domain is simply represented by a logical node [MuG97]. In the second approach, lower level domains are grouped together to form higher level domains. The most common ways to aggregate topology in this case are full mesh and symmetric star [Lee95]. In the full mesh approach, a sub-domain is represented by its border routers in its parent domain. The connection between two border routers in the sub-domain is represented by a logical link between them in the parent domain. The cost of the logical link is the minimum distance (i.e. hop count) between the border

routers. In the symmetric star representation, a sub-domain is represented by a single node in its parent domain. The full-mesh representation is much more accurate than the symmetric star representation and is therefore used for topology aggregation in the proposed protocol. The aggregated topologies of the example network from Figure 1 at level-2 and level-3 are shown in Figure 2.

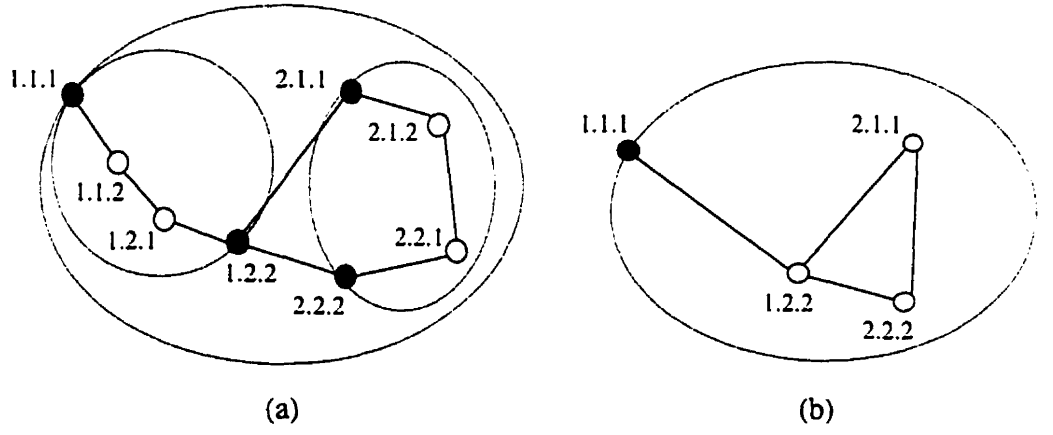

Figure 2 Aggregated topology: (a) level-2 domains; and (b) level-3 domain.

As the routing tables contain only topology information, they only need to be updated when there is a link failure or a new link is introduced into the network. The routing table can be constructed using a flooding algorithm. If the network graph is known, then actual flooding through the network is not necessary and the routing table can be constructed by executing the algorithm at each router. Every router in the network initiates a number of *GenerateRoutingTable* messages, one for each level it belongs to, and floods them through the network graph. As the messages are flooded through the network, they are processed by each router on their path. The pseudo-code to process the messages is presented in Figure 3. As shown in the pseudo-code, the routing table for level-(i-1) is

required to generate the routing table for level-i. Therefore, the algorithm has to be executed for all (i-1)-domains before it is executed for any i-domain.

```
// source is the router that initiated the current flooding message.
// distance is the hop count from the current router to the source along the path.
// preRouter is the neighboring router that forwarded the message to the current router.
// R_k (source, preRouter) is the entry in the k^th level routing table at the current router.
// k is the level of the domain in which flooding is being done.
// k-neighbor is a neighbor in the same k-domain as the current router.

GenerateRoutingTable (source, distance, preRouter, k)
{
        // all the entries in [R_k] are initialized to a number larger than
        // the maximum distance between any two routers in the network

        if (distance ≥ R_k (source, preRouter) )
                // message has traversed through a loop or a message through
                // a shorter path has already passed through the router.
                discard the message
        else
                R_k (source, preRouter) = distance
                for (∀ k-neighbor )
                        if (k-neighbor != preRouter)
                                preRouter = current router's address
                                if (k = 1)
                                        distance = distance + 1
                                else
                                        distance = distance + min{R_k-1(k-neighbor, j), ∀j}
                                end if
                                forward the message to the k-neighbor
                        end if
                end for
        end if
}
```

Figure 3 Pseudo-code for processing flooding messages to generate the routing table.

3.4 QoS State Metrics

As discussed earlier, QoS is defined as the ability of network elements (e.g. applications, hosts, and routers) to provide some level of assurance that the data traffic can meet certain service requirements. The most common service requirements are minimum bandwidth, maximum delay, maximum delay jitter (i.e. maximum variation of delay) or some other maximum cost associated with the data delivery. Bandwidth is a concave QoS metric, while delay, delay jitter and cost are additive QoS metrics [ChN98]. One QoS metric from each category (i.e. bandwidth and delay) is considered in the proposed protocol. The protocols for delay jitter and cost will be exactly the same as that for the delay.

All routers are assumed to keep up-to-date local QoS state metrics for all connecting links. *Bandwidth*(i, j) is the residual (unused) bandwidth of the link (i, j) and *delay*(i, j) is the channel delay in the link (i, j), including the propagation delay (*link delay*), the queueing delay, and protocol processing time (*node delay*). The QoS metrics are asymmetric over links (i.e. *bandwidth*$(i, j) \neq$ *bandwidth*(j, i) and *delay*$(i, j) \neq$ *delay*(j, i)). The QoS state metrics of a path $P = i \rightarrow j \rightarrow \ldots \rightarrow k \rightarrow l$ are defined as:

$$bandwidth(P) = \min\{bandwidth(i, j), \ \ldots, \ bandwidth(k, l)\}$$

$$delay(P) = delay(i, j) + \ldots + delay(k, l)$$

4 QHMRP: QoS-Aware Hierarchical Multicast Routing Protocol

4.1 Preliminary Remarks

The QoS-aware Hierarchical Multicast Routing Protocol (QHMRP) can be used for constructing both source-based and shared multicast trees. The protocol uses a controlled flooding algorithm, which relies on the local network states and routing tables available at each router. The routing tables (Section 3.3) contain topological information that needs to be updated only when a link fails or a new link is added to the network. Different messages used in the protocol are forwarded based on the algorithms implemented at local routers. The distributed nature of the algorithm and use of topology aggregation for routing tables provide scalability of the protocol. A detailed description and analysis of the protocol used for constructing shared multicast trees are presented in this chapter. A brief outline of the protocol for source-based trees is also given.

4.2 QHMRP for Constructing Shared Trees

Using the network and QoS models described in the last chapter, QHMRP may be used to create, join, and leave a shared multicast tree that can provide certain QoS guarantees. The different messages and data structures used to implement these functions are described below.

As mentioned earlier, the controllers of different domains in the network store information on multicast trees and facilitate the operation of QHMRP. The controllers of level-1 domains have lists of all the on-tree routers in their domain. All other higher level controllers have the controller address of their sub-domains having one or more on-tree routers. Therefore, if a multicast tree exists in the Internet, then there is at least one controller in every level of the hierarchy that is aware of the multicast tree.

4.2.1 Creating the Multicast Tree

The multicast tree is created when the first member (a host router) of the multicast group initiates the join process by sending a *JoinRequest* message to its parent controller. The format of this message is *JoinRequest*(*multicast*, *host*, *path*, *QoSType*, *QoSReq*). Here, *multicast* is the multicast group address, *host* is the host router's address, *path* is an array containing addresses of all the routers in the path of the *JoinRequest* message, *QoSType* specifies the type of service requested (e.g. bandwidth guarantee or delay guarantee, etc.) and *QoSReq* is the QoS requirement. When the host router initiates the *JoinRequest* message, the first entry in the array *path* is set to be the host router's address. The maximum number of entries in *path* is equal to the maximum number of levels in the hierarchy. If the controller receiving a *JoinRequest* message is not aware of the multicast tree, it appends its own address to the array of addresses in *path* and forwards the *JoinRequest* message to its parent controller. If the requested multicast tree does not exist in the network, the *JoinRequest* message will arrive at the controller of the highest domain, which is not aware of the multicast tree. In this case, the highest level controller sends a *CreateTree* message towards the *host*. The pseudo-code for processing

JoinRequest messages is shown in Figure 4. The Flooding message used in the pseudo-code is discussed later in Section 4.3.4.

```
JoinRequest(multicast, host, path, QoSType, QoSReq)

{
        if (current router is on the tree)
                send Flooding messages towards the host
        else if (on-tree controllers or routers exist in the domain)
                forward JoinRequest message to all on-tree controller and routers
        else if (current router is the highest level controller)
                send CreateTree message to the last address in the array path
                discard the JoinRequest message
        else
                append the current router's address to the array path
                forward the JoinRequest message to the current router's parent controller
        end if
}
```

Figure 4 Pseudo-code for processing the *JoinRequest* message.

The *CreateTree* message travels a path opposite to that traversed by the *JoinRequest* message and reaches the host router. Upon receiving the *CreateTree* message, the host router creates the tree having it-self as the only on-tree router and sends an *UpdateTree* message to its parent controller. The *UpdateTree* message updates the tree information at the controller and is forwarded towards the higher level controllers. The pseudo-code for the *CeateTree* and *UpdateTree* messages are shown in Figure 5 and Figure 6, respectively. The variable *router* in the *UpdateTree* message is the address of the router that sends the message. All other variables in these messages are same as those used in the *JoinRequest* message.

Every router has a boolean variable called *treeStatus(multicast)* to keep track of whether the router is on a specific multicast tree or not. This variable is initialized to FALSE and if the router becomes part of a multicast tree, then it is set to TRUE. Every controller in the network also has an array called *onTreeRouters(multicast)* that stores addresses of all on-tree routers within its domain and controller addresses of all sub-domains which have on-tree routers. As a *UpdateTree* message arrives at a controller, the address of the router that sent the message is added to the array *onTreeRouters(multicast)*.

```
CreateTree(multicast, host, path) {
        if (the current router is the host)
                treeStatus(multicast) = TRUE
                router = current route's address
                send an UpdateTree message to parent controller
        else
                remove the last address from the array path
                forward the CreateTree message to the router in the last entry of path
        end if
}
```

Figure 5 Pseudo-code for processing the *CreateTree* message.

```
UpdateTree(multicast, router) {
        append router to the array OnTreeRouters(multicast)
        if ( (the current router is the highest level controller) or
            (the current router is a controller having on-tree routers) )
                discard the UpdateTree message
        else
                router = current router's address
                forward the UpdateTree message to the parent controller
        end if
}
```

Figure 6 Pseudo-code for processing the *UpdateTree* message.

4.2.2 Joining the Multicast Tree

When a host router wants to join a multicast group, it sends a *JoinRequest* message to its parent controller. If the *JoinRequest* message arrives at a controller that is aware of the multicast tree, then the controller forwards the *JoinRequest* message to all the on-tree routers or controllers of the sub-domains having on-tree routers. Otherwise, the controller forwards the *JoinRequest* message to its parent controller (Figure 4). When the *JoinRequest* message arrives at an on-tree router, the router initiates a *Flooding* message. This message is flooded towards the host router by sending it to all neighbors, which in turn forward the message to their neighbors except the one that sent the message. The process of flooding from on-tree routers towards the host router is called *reverse flooding*.

During reverse flooding, the QoS provided by two *Flooding* messages coming from neighboring on-tree routers may not differ substantially. Therefore, the message overhead can be reduced by selecting fewer on-tree routers to flood towards the host router. Some of the concepts from QoSMIC [FaB98] may be useful for this purpose. This issue, however, is beyond the scope of the current thesis.

To reduce message overhead during reverse flooding, the messages are forwarded only in those directions that satisfy certain forwarding conditions. The forwarding conditions are selected to eliminate those messages that will not participate in establishing a feasible path between the host router and the multicast tree. Three different forwarding conditions are proposed for QHMRP. The *hierarchical condition* allows flooding only within a sub-

domain that contains both the current router and the host router. The hierarchical naming scheme presented in Chapter 3 is used to implement this condition. The second forwarding condition, the *topological condition*, uses the routing table to flood messages only along those links, which lead to the host router. The third condition, the *QoS condition*, allows flooding only along those directions, which satisfy the QoS requirement. The details of these forwarding conditions are discussed in Section 4.3. As the messages are flooded through the network, they estimate the QoS metric of the path (i.e. *QoSPath*) traversed by the message and implement the QoS forwarding condition.

Each router has a data structure defined by *F(multicast, host).ForwardStatus*, *F(multicast, host).QoSPath*, and *F(multicast, host).PreRouter* to store information on reverse flooding that is required for establishing connection during the join process. Here, *F(multicast, host).PreRouter* is the address of the neighboring router that sent the flooding message which has already been forwarded by the current router, *F(multicast, host).ForwardStatus* is a boolean variable that shows whether the flooding message has been forwarded by the current router or not, and *F(multicast, host).QoSPath* is the *QoSPath* of the most recent message that has been forwarded by the router and is initialized to zero. The variable *F(multicast, host).ForwardStatus* has a default value FALSE and is set to TRUE when a flooding message from the multicast tree is forwarded towards the host router.

Because of the QoS forwarding condition, all the *Flooding* messages that arrive at the host router satisfy the QoS requirement. After receiving the first *Flooding* message, the host router sends a *Join* message along a path that is opposite to the path followed by the

Flooding message. The variables in the data structure *F(multicast, host)* at each router are used to find the path for the *Join* message. All the routers in the path of the *Join* message become part of the branch that connects the host router with the tree. When a router receives a *Join* message, it reserves the resources for the multicast tree, updates the tree information at the current router, sends an *UpdateTree* message to its parent controller for updating the multicast tree information, and forwards the message to *F(multicast, host).PreRouter.*

Even though a router has sufficient resources to meet the QoS requirement when it forwards the *Flooding* message, it may not have the required resources to reserve while processing the *Join* message. This problem can be avoided by reserving resources while forwarding the *Flooding* message and releasing the resource if it is not used before a certain specified time. This approach will unnecessarily reserve more resources than required for a certain period of time. The issue of resource availability while processing *Join* messages is also outside the scope of this thesis.

Every router has an array *treeNeighbour(multicast)* that stores the addresses of all the on-tree neighboring routers. The variable *PreRouter* contains the address of the router that forwarded the *Join* message. When a router receives a *Join* message it adds *PreRouter* to the array *treeNeighbour(multicast)*. Similarly, when a router forwards a *Join* message, it adds *F(multicast, host).PreRouter* to the array *treeNeighbour(multicast)*. Since *Join* message follows the reverse path of a *Flooding* message, it will eventually arrive at one

of the on-tree routers, and will complete the join process. The pseudo-code for processing the *Join* message is presented in Figure 7.

```
Join(multicast, host, PreRouter, QoSType, QoSReq)
{
        append PreRouter to treeNeighbour(multicast)
        if ( treeStatus(multicast) = TRUE )
                discard the Join message
        else
                treeStatus(multicast) = TRUE
                reserve resources on the link to F(multicast, host).PreRouter
                append F(multicast, host).PreRouter to treeNeighbour(multicast)
                PreRouter = current router's address
                send Join message to F(multicast, host).PreRouter
                if (current router is not the highest level controller)
                        send updateTree message to current router's parent controller
                end if
        end if
}
```

Figure 7 Pseudo-code for processing the *Join* message.

4.2.3 Leaving the Tree

When a host wants to leave the multicast tree, it sends a *Leave* message to the host router. The host router disconnects the host from the multicast tree. Then, if the router does not have any other host and is therefore now a leaf node of the multicast tree, then it forwards the *Leave* message to the neighboring on-tree router. This process continues until the forwarding condition for the *Leave* message is violated by an on-tree router.

4.3 Reverse Flooding

The main limitation of any protocol using a flooding algorithm is the high message overhead. This can be reduced by eliminating flooding in those directions, which will not reach the flooding destination (e.g. the host router in the proposed approach). The flat unicast protocol in [ChN98] uses a QoS forwarding condition to reduce message overhead where messages are flooded only along those links, which satisfy the QoS requirement. In addition to using the QoS forwarding condition, two additional conditions, the hierarchical and topological forwarding conditions, are proposed for QHMRP. The details of these conditions are described below. The forwarding conditions at a router decide whether the *Flooding* message should be forwarded along a connecting link or not.

4.3.1 The Hierarchical Forwarding Condition

This forwarding condition limits flooding within the lowest level domain that contains the host router and some on-tree routers. The domain in which reverse flooding is done is called the *flooding domain.* The forwarding condition is implemented by a distributed algorithm, which uses the addresses of the current router and the host router. It allows flooding only within the lowest level domain that contains both the current router and the host router. This domain is obtained by comparing the addresses expressed using the naming scheme discussed in Chapter 3. Let the addresses of the current router and the host router be $(i^c_{L-1} i^c_{L-2} \cdots i^c_2 i^c_1 i^c_0)$ and $(i^h_{L-1} i^h_{L-2} \cdots i^h_2 i^h_1 i^h_0)$, respectively. Let j be the level of the lowest level domain that contains both the current router and the host router.

Then, $i_x^c = i_x^h,\ \forall x > (j-1)$ and $i_{j-1}^c \neq i_{j-1}^h$. This means, *Flooding* message should be forwarded to all neighboring routers, which are in the same k-level ($\forall k \leq j$) domains as the current router. Two routers a and b belong to the same k-level domain if $i_x^a = i_x^b,\ \forall x > (k-1)$.

4.3.2 The Topological Forwarding Condition

The topological forwarding condition uses the routing tables to forward *Flooding* messages only towards those neighboring routers, which are on a path to the host router. Since the network is organized in a hierarchical structure and full-mesh topology aggregation is used to represent network topology, the routing tables at every router may not have an entry for the host router. Let j be the level of the lowest level domain that contains both the current router and the host router, i.e. $i_x^c = i_x^h,\ \forall x > (j-1)$ and $i_{j-1}^c \neq i_{j-1}^h$. If the current router belongs to a j-domain, then the *Flooding* message should be sent towards the border routers of the (j–1)-domain that contains the host router. If a j-domain router $(i_{L-1}\, i_{L-2} \cdots i_2\, i_1\, i_0)$ also belongs to the (j–1)-domain that contains the host router, i.e. $i_x = i_x^h,\ x > (j-2)$, then the router is a border router of the (j–1)-domain containing the host router. If the current router belongs to a k-domain, where $k < j$, then the host router is not in any sub-domain of the k-domain containing the current router and the *Flooding* message should be sent out of the current domain. This can be done by flooding towards all neighboring routers in domains below the j-domain.

The above approach floods towards all neighbors, which are on the path to the host router. Since the routing table contains the distance (i.e. hop count) from the current router to all other routers in the domain, the message overhead can further be reduced by flooding towards a limited number of neighbors which connect to the host router by shorter paths.

4.3.3 The QoS Forwarding Condition

Any message that does not satisfy the QoS requirement along its path will be discarded by the host router. Therefore flooding message overhead can be minimized by allowing flooding only along directions, which satisfy the QoS requirement. For implementing QoS routing, the *Flooding* messages collect the QoS state metric of the path they follow on their way to the host router. Let i and j be two neighboring routers, P_i be the path followed by the *Flooding* message from its origin to node-i and P_j be the path to node-j which passes through node-i. The QoS state metric of the path can be recursively defined as:

$$bandwidth(P_j) = \min \{ bandwidth(i, j), bandwidth(P_i) \}$$

$$delay(P_j) = delay(i, j) + delay(P_i)$$

where $bandwidth(P_k)$ and $delay(P_k)$ are the bandwidth and delay, respectively, of the path P_k , $bandwidth(i, j)$ is the residual (unused) bandwidth of the link(i, j), and $delay(i, j)$ is the channel delay in the link(i, j). The channel delay includes the propagation delay (*link delay*), the queueing delay, and the protocol processing time (*node delay*). A *Flooding* message at node-i is forwarded towards node-j, if $bandwidth(i, j) >$ (bandwidth requirement) or $delay(P_j) <$ (delay requirement). If the bandwidth requirement is satisfied

for each link in a path, then it is satisfied for the whole path. Therefore, the bandwidth forwarding condition does not check the QoS metric of the path.

4.3.4 The Flooding Algorithm

A *Flooding* message is forwarded to a neighbor only if it satisfies all the three forwarding conditions discussed. However, satisfying forwarding conditions does not guarantee that there will be no loops during flooding. Loops can be prevented by using different flooding techniques which decide whether a router should forward the *Flooding* messages or not. The flooding technique-1 allows only one *Flooding* message for a particular (*multicast*, *host*) pair to pass through each router and, therefore, prevents loops. This technique is implemented by setting the variable *F*(*multicast*, *host*).*ForwardStatus* to TRUE after forwarding a *Flooding* message. This variable can be used to discard any future message for the same (*multicast*, *host*) pair.

For the delay QoS requirement, the flooding technique-1 may not find an existing feasible path [ChN98]. This problem occurs when a *Flooding* message with higher path delay arrives at a router before *Flooding* messages with lower path delay. In this case, the message with higher path delay gets forwarded the router. If the forwarded message fails to satisfy the QoS requirement later, it will not reach the host router and the feasible path may be detected. In the context of unicast routing, this problem was solved in [ChN98] by delaying the *Flooding* message at each router by Δt, where Δt = (node delay). The introduction of delay at each router guarantees that the messages with lower path delay arrive at routers before messages with higher path delay. However, this technique

increases the connection time for joining a multicast group. This technique is referred to as the flooding technique-2.

To reduce the connection time while increasing the chance of finding a feasible path, two additional flooding techniques are proposed. In the first alternate approach (i.e. flooding technique-3), if the difference between the *QoSPath* of the current message and that of a previously forwarded message is more than ΔD, where $\Delta D = \alpha \times QoSReq$ and $0 < \alpha < 1$, then the message is forwarded. Here, *QoSPath* is the QoS metric of the path followed by the *Flooding* message up to the current router. This technique increases the chance of finding an existing feasible path by allowing second and subsequent *Flooding* messages to pass through a router if their path delay is better than the previously forwarded *Flooding* message. The message overhead of this technique is higher than that of the technique-1 and technique-2. The *QoSPath* of a message that has gone through a loop is higher than that of the same message when it was forwarded by the router for the first time. Therefore, this technique will reject any message that has gone through a loop.

The second alternate approach (i.e. flooding technique-4) is a combination of the technique-3 and the approach used in [ChN98] (i.e. technique-2). In this case, if $QoSReq > \Delta D > (\alpha \times QoSReq)$, then the message is forwarded after a delay that is less than the node delay (i.e. $\Delta t <$ node delay). This technique has lower message overhead than technique-3 and also has lower connection time than technique-2. The algorithm to process the *Flooding* message is presented in Figure 8.

```
Flooding (multicast, host, PreRouter, QoSType, QoSReq, QoSPath)
{
        if ( (QoSType = bandwidth) and (F(multicast, host).ForwardStatus = FALSE) )
                ForwardMessage = TRUE
                Δt = 0
        else if ( (QoSType = delay) and (F(multicast, host).QoSPath – QoSPath) > ΔD)
                ForwardMessage = TRUE
        else
                ForwardMessage = FALSE
        end if

        if (ForwardMessage = TRUE)
                for (every neighboring router)
                        if (all three forwarding conditions are satisfied)
                                F(multicast, host).QoSPath = QoSPath
                                F(multicast, host).PreRouter = PreRouter
                                F(multicast, host).ForwardStatus = TRUE
                                PreRouter = current router's address
                                if (QoSType = delay)
                                        QoSPath = QoSPath + node delay
                                end if
                                send Flooding message to the neighbor after time Δt
                        end if
                end for
        end if
}
```

Figure 8 Pseudo-code for processing the *Flooding* message.

All four flooding techniques for delay QoS requirement are implemented by using the last technique with appropriate values for ΔD and Δt as given in Table 1. When ΔD is equal to the *QoSReq*, only the first message arriving at a router is forwarded.

Table 1 Parameters for specifying different flooding techniques.

	ΔD	Δt	Comments
Technique – 1	$\Delta D_1 = QoSReq$	0	Allows one message per link without any extra delay
Technique – 2	$\Delta D_2 = QoSReq$	node delay	Allows one message per link with delay equal to node delay
Technique – 3	$\Delta D_3 = (\alpha \times QoSReq), 0 < \alpha < 1$	0	Allows more than one message per link without any extra delay
Technique – 4	$QoSReq > \Delta D_4 > (\alpha \times QoSReq)$	< node delay	Allows more than one message per link with delay less than node delay

4.4 Analysis of QHMRP

Connection time and message overhead are two important performance metrics for routing algorithms. They have a direct impact on the applicability of the algorithm to real world problems. This section presents an analysis of the worst-case connection time and message overhead of QHMRP. Analysis is also done to show that the proposed protocol does not form loops during reverse flooding.

4.4.1 Complexity Analysis

As discussed in the last section, there are three steps involved in establishing a connection between a host router and a multicast tree. They are unicasting a *JoinRequest* message from the host router to on-tree routers via controllers, flooding messages from on-tree routers towards the host router, and sending a *Join* message from the host router to an on-tree router. Let the time taken by the *JoinRequest* and *Join* messages to traverse a link

including the buffering and processing time at nodes be one unit of time. Then the time taken by the *JoinRequest* and *Join* messages together is $O(l_1+l_2)$, where l_1 is the length of the path followed by the *JoinRequest* message and l_2 is the length of the *Join* message path.

The path of the *Join* message is opposite of the path followed by the *Flooding* message that is used to initiate the *Join* message. Therefore, the time taken by the *Flooding* message is the sum of the delays at each router in the path of the *Join* message, i.e. $\sum_{i=1}^{l_2}(\Delta t)_i$. Thus, the time taken by the *Flooding* message depends on the flooding technique used (see Table 1). Technique-2 has the highest connection time while the lowest connection time is provided by technique-1. In all the cases, the time required by the *Flooding* message in $O(l_2)$. Therefore, the total connection time for the protocol is $O(l_1+2l_2)$.

To estimate the message overhead, sending a message over a link is counted as one message. The number of messages per join request depends on the number of on-tree routers, size of the flooding domain, QoS requirement, and the flooding technique used. The number of *JoinRequest* and *Join* messages per join request is l_1+l_2. For bandwidth requirement and delay requirement with flooding techniques-1 and 2, the protocol sends at most one *Flooding* message per link for each (*multicast, host*) pair. The total number of *Flooding* messages is thus bounded by e where, e is the number of links in the flooding domain. Therefore, the worst-case message overhead is $O(e+l_1+l_2)$.

For flooding techniques-3 and 4, the number of flooding messages sent over a link depends on the value of ΔD. Compared to technique-1, these methods have additional message overhead that depends on the total number of links (N_p) in all possible paths between the on-tree routers and the host router within the flooding domain. Thus, the worst-case message overhead per connection request can be expressed as $O(N_p+e+l_1+l_2)$. The average message overhead will be substantially smaller than the worst-case overhead when different forwarding conditions are used. This is conformed by the simulation results presented in Chapter 5.

4.4.2 Preventing Loops during Flooding

If bandwidth requirement or delay requirement with techniques-1 and 2 is used, then a single *Flooding* message for a particular (*multicast*, *host*) pair is allowed to pass through a router. Therefore, all subsequent messages for the same (*multicast*, *host*) pair are discarded, and the formation of loops during flooding is prevented.

In techniques-3 and 4 for delay requirement, the second and subsequent *Flooding* messages will be discarded if the path delay is larger than that of the previously forwarded message. The path delay of a message after it has gone through a loop will be higher than that of the message when it was forwarded for the first time. Therefore, the forwarding condition will prevent a message from passing through the same router more than once and therefore the formation of loops is prevented.

4.5 QHMRP for Constructing Source-Based Trees

The earlier sections in this chapter described the details of QHMRP for constructing a shared multicast tree. This section briefly outlines the procedure for generating a source-based tree using QHMRP. The detailed description and analysis of this protocol is not included in this thesis. When a source wants to construct a multicast tree, it sends *Flooding* messages towards the receivers of the multicast group. The messages can be flooded through the network using the forwarding conditions and flooding techniques described in Section 4.3. The hierarchical forwarding condition makes sure that the flooding is done within the flooding domain, which is the lowest level domain that contains the source and all destination routers. When a *Flooding* message reaches a receiver, a *Join* message is sent towards the source router along a path that is opposite to the *Flooding* message path. When the *Join* message reaches an on-tree router or the source router, the connection is complete.

Once the source-based tree is constructed, new receivers can join or existing receivers can leave the tree using the QHMRP for a shared multicast tree described in Section 4.2. Therefore, the source-based tree need not be constructed again for any change in the membership of the multicast group.

5 SIMULATION

5.1 Preliminary Remarks

The effectiveness and performance of QHMRP was assessed by simulation using PARSEC (PARallel Simulation Environment for Complex Systems), a C-based discrete-event simulation environment [Mey98]. The proposed routing protocol for constructing a shared multicast tree was implemented for flat as well as hierarchical networks. The performance of QHMRP and the flat routing protocol were compared to assess the advantages of the hierarchical scheme. The next section describes the performance measures, network model, and parameters used in the simulation. This is followed by the simulation results and discussions.

5.2 Simulation Model

The performance metrics used to assess the performance of QHMRP are success ratio, average message overhead, and average connection time. These are defined as:

$$\text{success ratio} := \frac{\text{number of hosts accepted}}{\text{total number of join requests}}$$

$$\text{average message overhead} = \frac{\text{total number of messages sent}}{\text{total number of join requests}}$$

$$\text{average connection time} = \frac{\text{total connection time of all successful join requests}}{\text{total number of join requests}}$$

The connection time of a successful join request is the time difference between the start of a *JoinRequest* message and end of the *Join* message which completes the join process.

For calculating the message overhead, sending a *Flooding* message over a link is counted as one message.

The network used in the simulation consists of 56 nodes and 77 links, and is presented in Figure 9 [HaZ99]. The figure also shows the domains and sub-domains used to organize the network into the hierarchical structure. The controller for each domain is specified in the simulation. All the routers are given unique IDs according to the naming scheme described in Section 3.2. The bandwidth capabilities of all links in 1-domain are assumed to be 155 Mbps and all other links are 622 Mbps. The background traffic load on each link is randomly generated in the range of [0, 155] Mbps. The control-delay (i.e. delay in processing and forwarding control messages) of each link is randomly generated from [0, 60] ms. This delay is used to model the processing and transmission delay of each control message used in the protocol. Similarly, the node-delay, i.e. delay for multicast data processing and transmission, of each link is randomly generated from [0, 200] ms. The same network with all 56 routers in a single domain is used to simulate the flat routing protocol.

5.3 Simulation Results

The proposed protocol was implemented for hierarchical and flat network topologies for both bandwidth and delay QoS requirements. Two forwarding conditions, the hierarchical and QoS conditions, were implemented in the simulation. All four flooding techniques (refer to Table 1) for the delay QoS requirement presented in Section 4.3.4 were also

implemented. Performance measures are presented for different values of link traffic loads, node-delay, and QoS requirements.

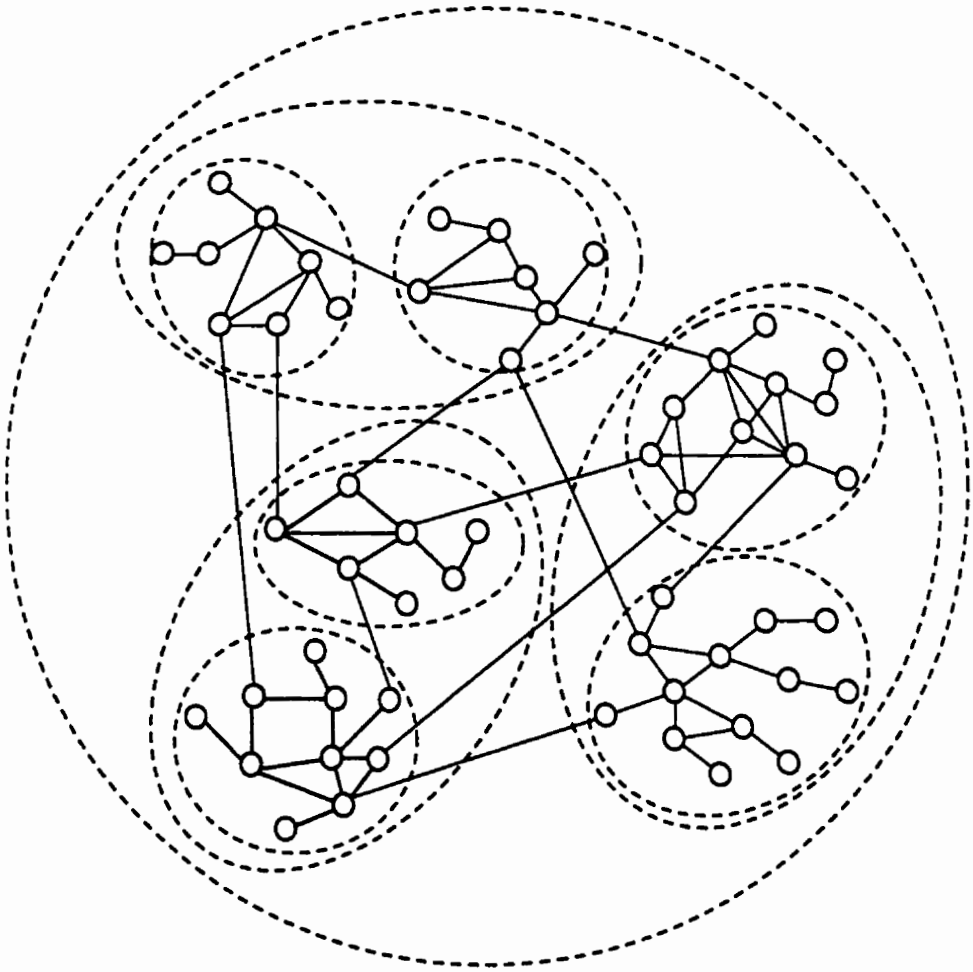

Figure 9 Network topology used in simulation.

Each point in the graphs presented below is the results of 150 simulation runs. In each simulation run, a random multicast tree with a specified number of nodes (5, 10, 15 or 28 i.e. half of the total number of routers) was generated. A host router was randomly selected from the off-tree nodes. The bandwidth and delay QoS requirements were

randomly selected from the range [1, 15] Mbps and [400, 600] ms, respectively. The performance metrics for each simulation run were calculated and the average values of performance metrics for all 150 runs were estimated.

Simulation results are presented separately for each performance metric. The performance results are primarily determined by the size of the flooding domain and the distance between the host routers and on-tree routers. The size of the multicast tree decides these parameters. For larger multicast trees, the size of the flooding domain, which contains the host router and an on-tree router, will be smaller and vice-versa. Similarly, as the size of the tree increases, the average distance between the host router and the nearest on-tree router decreases. The simulation results and discussion are presented below.

5.3.1 Success Ratio

The success ratio for different node delays and tree sizes for simulations with delay QoS requirements are presented in Figure 10. As the same simulation parameters are used for the flat as well as hierarchical routings, the success ratio has similar behavior in both cases. The path delay of *Flooding* messages increases with the node delay. Therefore, for a given tree size, the number of messages rejected by the QoS forwarding condition increases and, hence, the success ratio decreases with an increase in the node delay. Since the average path length of *Flooding* messages is higher for fewer on-tree routers, this effect is more prominent for 5-node trees than in the other cases (Figure 10).

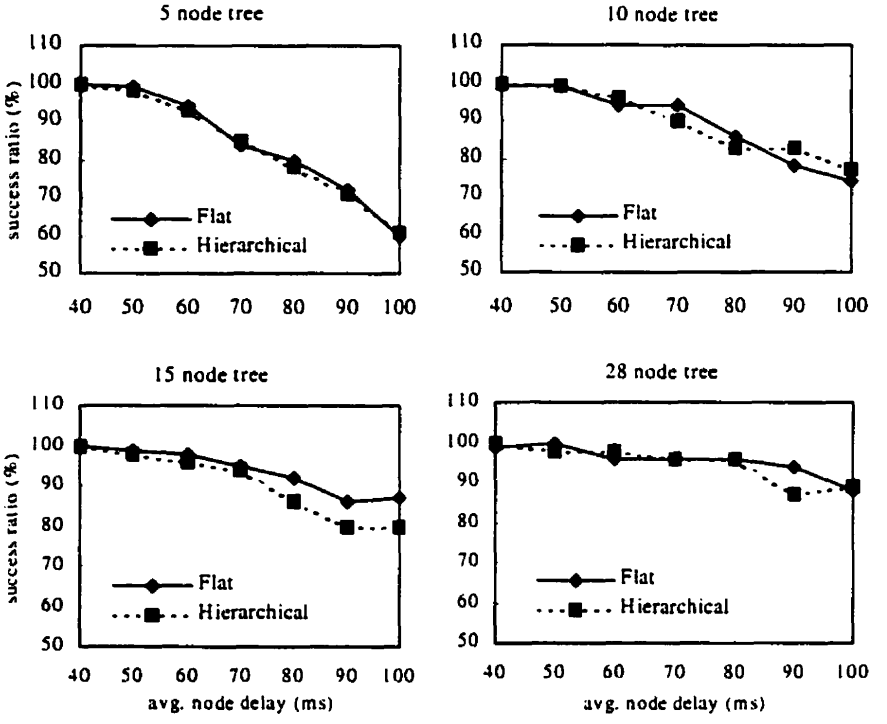

Figure 10 Success ratios for different node delays and tree sizes.

The success ratio for different delay requirements is presented in Figure 11. As in Figure 10, the performance of hierarchical and flat routing protocols are similar. For a given tree size, the number of *Flooding* messages rejected by the QoS forwarding conditions decreases and, hence, success ratio increases with an increase in the delay requirement. The average path length of *Flooding* messages decrease with an increase in tree size. Therefore, for a given node delay, the path delay and the number of messages rejected by the QoS forwarding conditions decreases and, hence, the success ratio increases with an increase in tree size (Figure 11).

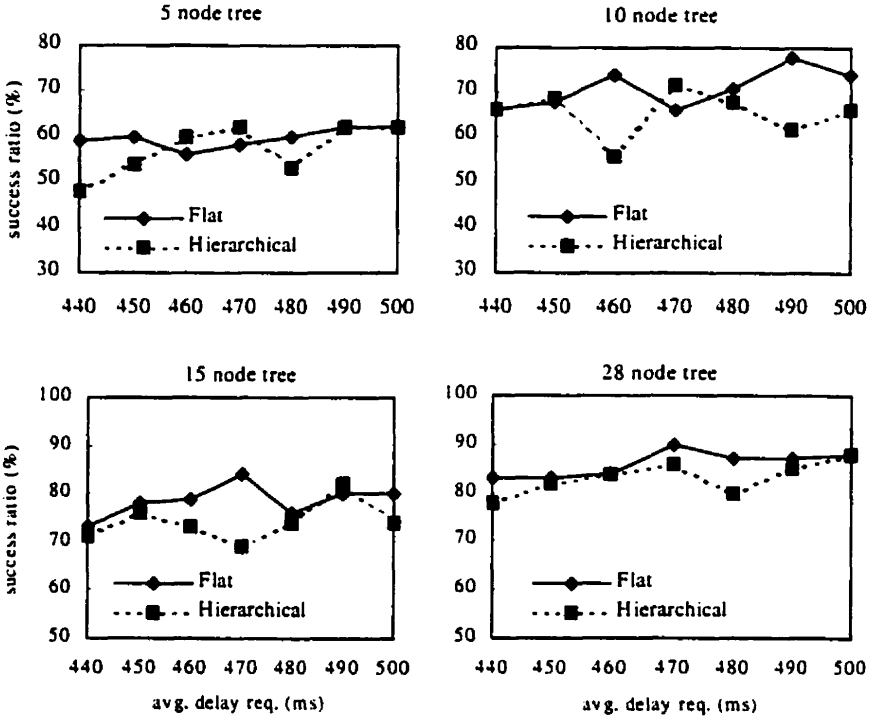

Figure 11 Success ratio for different delay requirements.

The success ratio for the bandwidth QoS requirement is not affected by the tree size. In this case, the success ratio decreases with an increase in average traffic load or bandwidth requirement (Figure 12).

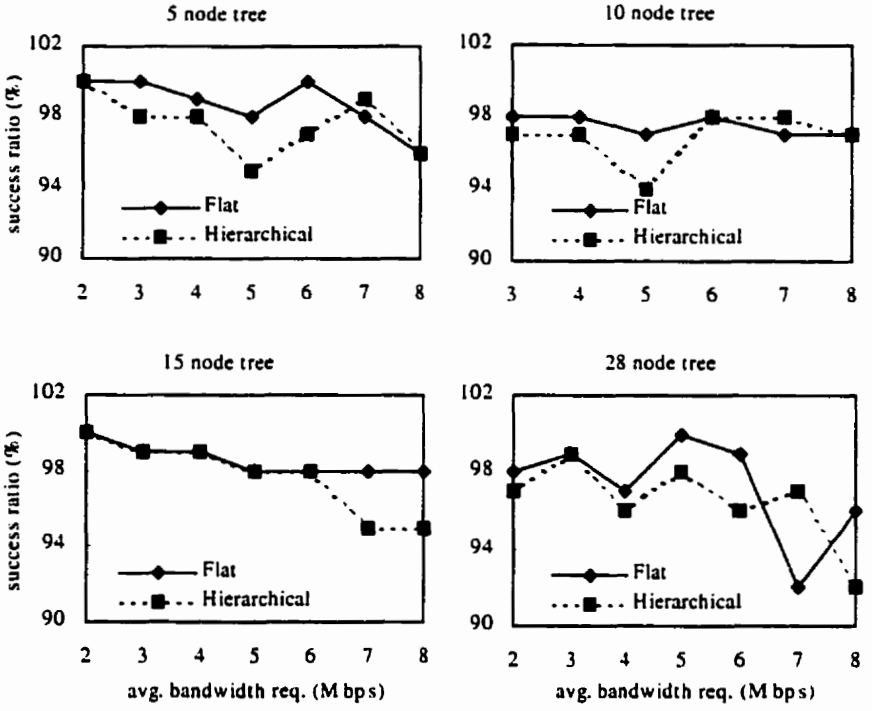

Figure 12 Success ratio for different bandwidth requirements.

5.3.2 Message Overhead

The effectiveness of the protocol mainly depends on the message overhead during reverse flooding. Therefore, the results presented in this section focus on the reverse flooding and omit the message overhead due to the *JoinRequest* and *Join* messages. Since it is difficult to quantify the path length for *JoinRequest* and *Join* messages, this approach allows comparison of the worst case message overhead estimated in Section 4.4.1 with the simulation results. Flooding technique-1 (Table 1) is used for the simulations presented in this section.

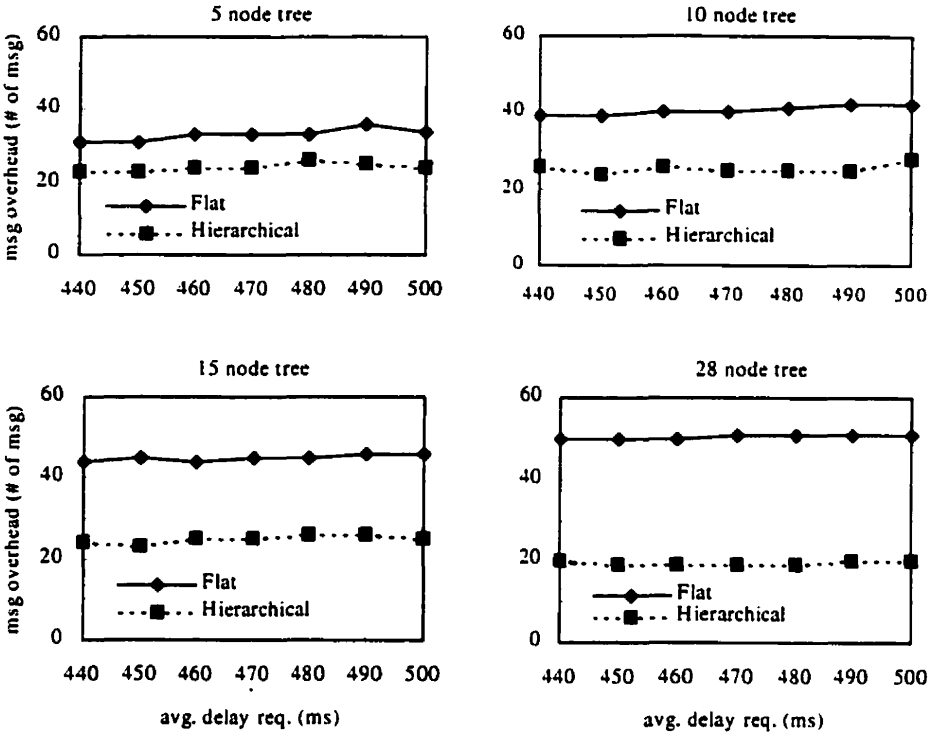

Figure 13 Message overhead for different delay QoS requirements.

The message overhead (i.e. number of messages) for the delay and bandwidth QoS requirements are presented in Figure 13 and Figure 14. The estimated worst-case message overhead is 77 for flat routing and less than 77 for hierarchical routing. As shown in the following figures, the average message overhead is significantly smaller than the worst case numbers. The advantage of hierarchical routing in terms of lower message overhead as compared to the flat routing scheme can be clearly seen from the figures for both the delay and bandwidth QoS requirements. This is mainly because of the smaller flooding domain in hierarchical routing as compared to flat routing.

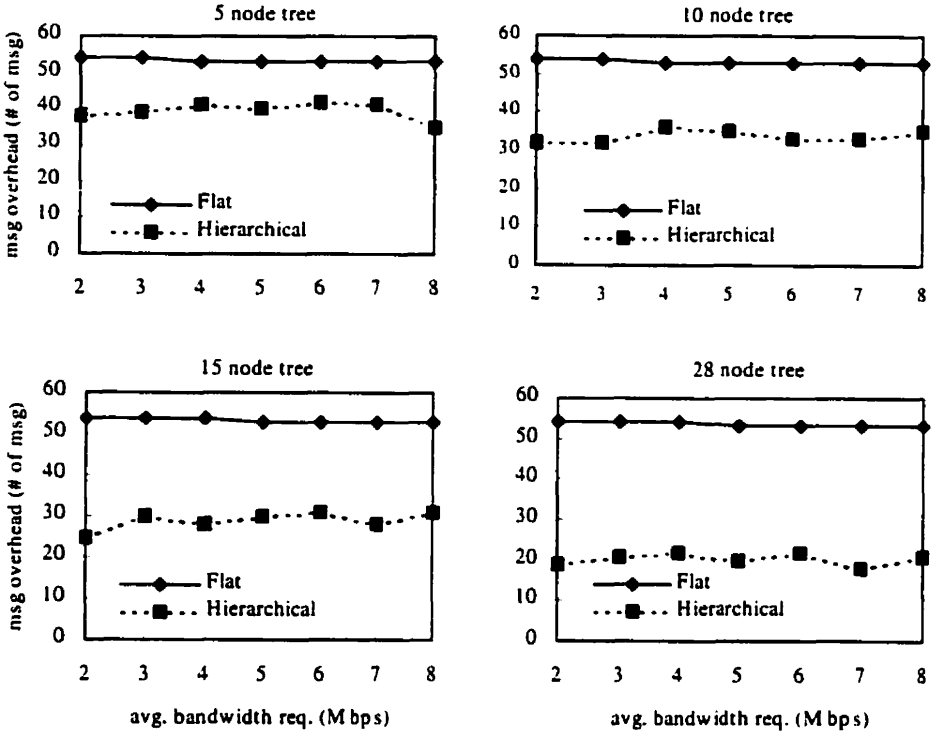

Figure 14 Message overhead for different bandwidth QoS requirements.

The path length and, hence, the path delay of *Flooding* messages increases with a decrease in the tree size. Therefore, for the delay QoS requirement, more messages are discarded and message overhead is lower with fewer on-tree nodes (see Figure 13). For the bandwidth QoS requirement, the QoS forwarding condition depends on the available bandwidth of each link. Therefore, the message overhead is independent of the path length and is primarily governed by the size of the flooding domain. For flat routing, the size of the flooding domain is independent of the tree size and, therefore, has similar message overhead for all cases. However, for hierarchical routing, the size of the flooding

domain and the message overhead decrease with an increase in the tree size (see Figure 14).

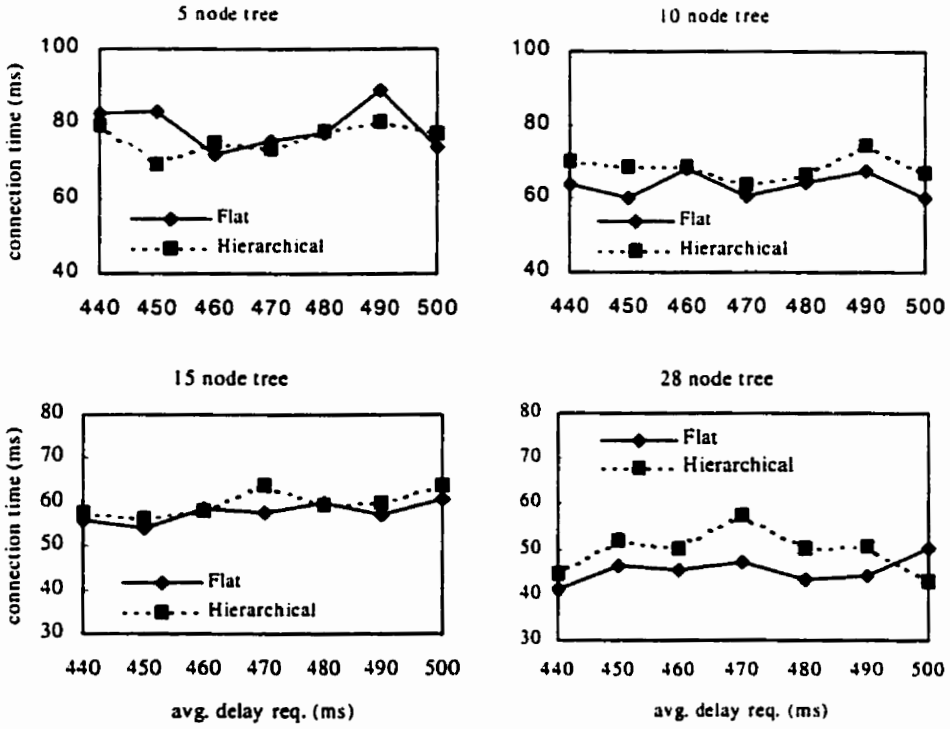

Figure 15 Connection time for different delay QoS requirements.

5.3.3 Connection Time

The connection time presented in Figure 15 is for a simulation with the delay QoS requirement and using flooding technique-1. As the connection time is decided by the path length of different messages, it is similar for both hierarchical and flat routing schemes. However, as the number of on-tree routers increases, the average path length of different messages decreases and, hence, the connection time is less for larger trees

(Figure 15). Similar behavior is also observed for the bandwidth QoS requirement (Figure 16).

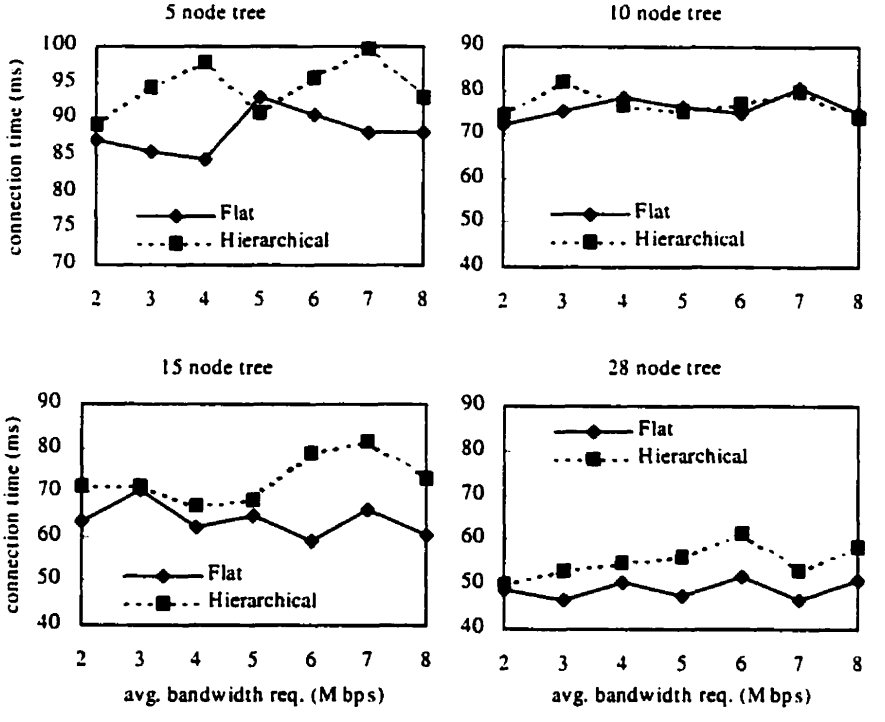

Figure 16 Connection time for different bandwidth QoS requirements.

5.3.4 Comparison of Flooding Techniques

The simulation study did not explicitly model the scenario which could lead to difficulty in finding a feasible path when flooding technique-1 is used. Therefore, all the flooding techniques have similar behavior for success ratio and message overhead. However, they have noticeable differences for the connection times (see Figure 17). Flooding techniques-1 and 3 do not introduce any additional delay (i.e. $\Delta t = 0$) during reverse

flooding. Therefore, they have similar connection times, which are smaller than the other flooding methods. Technique-2 has the highest Δt and, therefore, has the highest connection time. For techniques-2 and 4, the connection time is the sum of the node delays of all routers in the path. Therefore, it increases with an increase in the node delay (refer to Figure 17).

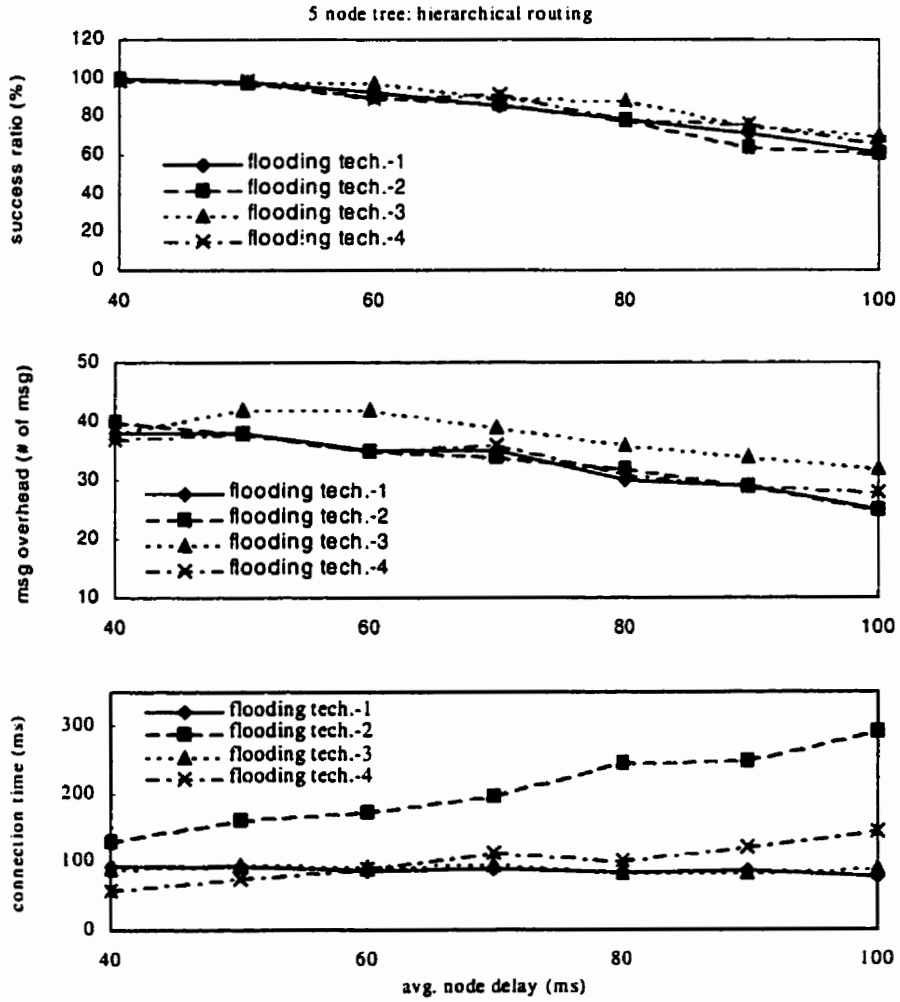

Figure 17 Connection time for different flooding techniques.

6 CONCLUSIONS AND FUTURE WORK

6.1 Concluding Remarks

A novel protocol, QHMRP (QoS-aware hierarchical multicast routing protocol), is proposed for multicast routing in large IP networks. To achieve scalability, the network is divided into domains organized into an L-level hierarchy. One of the border routers of each domain is designated to be the domain controller. Every router in a domain is either aware of the domain controller or can identify it using a query/response Session Directory. The controllers have the addresses of their sub-domain controllers and on-tree routers in their domain, and they facilitate the construction of the multicast tree. They do not, however, participate in the multicast tree like the core router in CBT and the RP in PIM-SM. Therefore, controllers in QHMRP are not performance bottlenecks. QHMRP can be used for creating and maintaining both shared and source-based multicast trees. The detailed protocol for constructing a shared multicast tree is presented was in Section 4.2. A brief outline of QHMRP for source-based trees was also presented in Section 4.5.

For creating or joining a multicast tree, the host router sends a *JoinRequest* message to its parent controller. If the multicast tree does not exist in the network, then this message gets forwarded to the highest level controller and a new tree with the host router as the only on-tree router is created. If the tree exists, then the *JoinRequest* message is

forwarded to all on-tree routers within the flooding domain. The flooding domain is the lowest level domain that contains the host router and some on-tree routers. After receiving a *JoinRequest* message, the on-tree routers initiate *Flooding* messages, which are flooded towards the host router. The process of flooding from the on-tree routers towards the host router is called *reverse flooding*. Hierarchical, topological, and QoS forwarding conditions are used to reduce the message overhead during reverse flooding. The hierarchical condition limits flooding within the flooding domain, the topological condition uses routing tables to flood only along links which lead to the host router, and the QoS condition forwards messages along a link only if the necessary QoS requirement(s) is satisfied.

Four flooding techniques were presented for the delay QoS requirement to prevent loops during reverse flooding and guarantee that a feasible connection path is detected, if one exists. The first technique allows only one message to pass through a router and forwards the message as soon as the processing is completed (i.e. no delay). The second technique forwards only one message after a delay that is equal to the node delay. The third technique allows more than one message to be forwarded by a router without any delay if the second and subsequent messages are better than the previously forwarded message. The fourth technique is a combination of the second and third, where fewer messages than technique-3 are forwarded after a delay that is less than technique-2.

The protocol for constructing a shared tree was implemented for both flat and hierarchical networks. The feasibility and performance of the protocol were assessed by simulation.

Results show that the performance in terms of message overhead of the QHMRP is better than the flat routing protocol. The advantages and contributions of the proposed protocol are follows:

- This is the first QoS-based hierarchical multicast routing protocol that can be used to construct both shared and source-based trees.
- QHMRP provides end-to-end QoS guarantee for shared trees.
- Novel forwarding conditions, i.e. hierarchical and topological conditions have been proposed to reduce message overhead during reverse flooding.
- New flooding techniques have been proposed to control the reverse flooding so that a feasible path can be found, if one exists.
- QHMRP is a distributed algorithm, where the processing at each router is based only on local state information. The full mesh aggregation technique is used to reduce the size of the routing tables used in the protocol. In addition, the routing tables need to be updated only when a link fails or a new link is introduced into the network. Therefore, the protocol is quite scalable quite scalable compared to other flat routing schemes.

6.2 Future Work

The protocol presented and the simulations performed in this thesis have certain limitations. The following extensions to the work would improve the performance and/or the validity of the protocol.

- Only two forwarding conditions (the hierarchical and QoS conditions) were implemented in the simulation. The topological condition should also be implemented to further reduce the message overhead during reverse flooding.
- In the current implementation of QHMRP, all the on-tree routers within the flooding domain initiate *Flooding* messages. However, the performance of two *Flooding* messages coming from neighboring on-tree routers may not differ substantially. Therefore, message overhead could be reduced by selecting fewer on-tree routers to flood towards the host router. Some of the concepts from QoSMIC may be useful for this purpose.
- The simulation results presented in this thesis are based on a fixed network topology. The performance of the protocol could be better assessed by doing simulation using different network topologies generated by a random graph generator or using actual Internet topology data.
- Simulations could be done to reproduce the difficulty that can occur in flooding technique-1. This will allow more realistic comparison of different flooding techniques proposed here.
- The performance of QHMRP was compared with a protocol using reverse flooding in a flat network. This shows the advantage of the hierarchical routing compared to the flat protocol. To show that QHMRP is a practical protocol, it should also be compared with other protocols such as QMRP and QoSMIC.
- As outlined in Section 4.5, the proposed forwarding conditions and flooding techniques can be easily used to construct source-based multicast trees. A detailed

protocol and simulation for constructing source-based multicast trees should be developed to show the wider applicability of QHMRP.

Acronyms

ALVA	Area-based Link Vector Algorithm
AS	Autonomous System
BR	Border Routers
CBT	Core Based Tree
CP	Center Point
DVMRP	Distance Vector Multicast Routing Protocol
HPIM	Hierarchical Protocol Independent Multicast
IP	Internet Protocol
MOSPF	Multicast extension to Open Shortest Path First
MPLS	Multi Protocol Labeling Switch
OCBT	Ordered Core Based Tree
OSPF	Open Shortest Path First
PIM	Protocol Independent Multicast
PIM-DM	Protocol Independent Multicast - Dense Mode
PIM-SM	Protocol Independent Multicast - Sparse Mode
QHMRP	QoS-aware Hierarchical Multicast Routing Protocol
QMRP	QoS-aware Multicast Routing Protocol
QoS	Quality of Service
QoSMIC	QoS sensitive Multicast Internet protoCol
RIP	Routing Information Protocol

RPF	Reverse Path Forwarding
RPM	Reverse Path Multicasting
RP	Rendezvous Point
RSVP	Resource reSerVation Protocol
TCP	Transmission Control Protocol

References

[AlS95] C. Alaettinoglu, and A. U. Shankar, "The Viewserver Hierarchy for Interdomain Routing: Protocols and Evaluation," IEEE Journal on Selected Areas in Communications, Vol. 13, No. 8, October 1995, pp. 1396-1410.

[Bal97] A. Ballardie, "Core Based Tree (CBT Version 2) Multicast Routing – Protocol Specification," RFC 2189, IETF, September 1997.

[Bal97a] A. Ballardie, "Core Based Tree (CBT Version 2): Multicast Routing Architecture," RFC 2201, IETF, September 1997.

[BeG98] Behrens, J., and Garcia-Luna-Aceves, "Hierarchical Routing Using Link Vectors," *Proceedings of the IEEE INFOCOM'98*, San Francisco, CA, 1998.

[ChN98] S. Chen, and K. Nahrstedt, "Distributed Quality-of-Service Routing in High-Speed Networks Based on Selective Probing," *Proceedings of the Conference on Local Networks'98 (LCN'98)*.

[ChN98a] S. Chen, and K. Nahrstedt, "An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions," *IEEE Network Magazine: Special Issue on Transmission and Distribution of Digital Video*, November/December 1998, <http://cairo.cs.uiuc.edu/papers.html>.

[ChN00] S. Chen, K. Nahrstedt, and Y. Shavitt, "A QoS-Aware Multicast Routing Protocol," to appear in *Proceedings of IEEE INFOCOM'2000*, Tel-Aviv, Israel, March 26-30, 2000.

[DaM78] Y. K. Dalal, and R. M. Metcalfe, "Reverse Path Forwarding of Broadcast Packets," *Communications of the ACM*, Vol. 21, No. 12, December 1978, pp. 1040-1048.

[DeC90] S. Deering, and D. Cheriton, "Multicast Routing in Datagram Internetwork and Extended LANs," *ACM Transactions on Computer Systems*, Vol. 8, No. 2, May 1990, pp. 85-110.

[DiD97] C. Diot, W. Dabbous, and J. Crowcroft, "Multipoint Communications: A Survey of Protocols, Functions, and Mechanisms," *IEEE Journal on Selected Areas in Communications*, Vol. 15, No. 3, April 1997, pp. 277-290.

[EsF96] D. Estrin, D. Farinacci, A. Helmy, V. Jacobson, and L. Wei, "Protocol Independent Multicast (PIM), Dense Mode Protocol Specification," *Internet Draft*, (draft-ietf-idmr-PIM-DM-spec-01.ps), September 1996.

[EsF98] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei, "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification," RFC 2362, IETF, June 1998.

[FaB98] M. Faloutsos, A. Banerjea, and R. Pankaj, "QoSMIC: Quality of Service sensitive Multicast Internet protoCol," *ACM SIGCOMM'98*, Vancouver, Canada, September 1998.

[FlH98] E. Fleury, Y. Huang, and P. K. McKinley, "On the Performance and Feasibility of Multicast Core Selection Heuristics," *Proceeding of the ICCCN'98*, pp. 196-303.

[GuO97] R. Guerin and A. Orda, "QoS-based Routing in Networks with Inaccurate Information: Theory and Algorithms," *Proceedings of the IEEE INFOCOM'97*, April 1997, Kobe, Japan.

[HaC] M. Handley, and J. Crowcroft, "Hierarchical Protocol Independent Multicast (HPIM)," <ftp://cs.ucl.ac.uk/darpa/IDMR/hpim.ps>.

[HaJ96] M. Handley, and V. Jacobson, "SDP: Session Directory Protocol (draft 2.1)," *Internet Draft - Work in Progress*, February 1996.

[HaZ99] F. Hao and E. W. Zegura, "On Scalable QoS Routing: Performance Evaluation of Topology Aggregation," *Technical Report GIT-CC-99-04*, College of Computing, Georgia Tech, 1999.

[Hed88] C. Hedrick, "Routing Information Protocol," RFC 1058, IETF, June 1988.

[HuF98] Y. Huang, E. Fleury, and P. K. McKinley, "LCM: A Multicast Core Management Protocol for Link-State Routing Networks," *Proceedings of the IEEE International Conference on Communications (ICC)*, Atlanta, Georgia, U.S.A., June 1998.

[HwR92] F. K. Hwang, and D. S. Richards, "Steiner Tree Problems," *IEEE Networks*, Vol. 22, No. 1, January 1992, pp. 55-89.

[KoM81] L. Kou, G. Markowshy, and L. Berman, "A Fast Algorithm for Steiner Trees," *Acta Informatica*, Vol. 15, 1981, pp. 141-145.

[Lee95] W. C. Lee, "Topology Aggregation for Hierarchical Routing in ATM Networks," *Computer Communications Review*, Vol. 25, No. 2, April 1995, pp. 82-92.

[Mey98] A. Meyer, PARSEC User Manual, Release 1.1, August 1998, UCLA Parallel Computing Laboratory, <http://pcl.cs.ucla.edu>.

[MoV98] M. Montgomery, and G. de Veciana, "Hierarchical Source Routing Through Clouds," *Proceedings of the IEEE INFOCOM'98*, San Francisco, CA, 1998, pp. 685-692.

[Moy94] J. Moy, "Multicast Routing Extensions for OSPF," Communications of the ACM, Vol. 37, No. 8, August 1994, pp. 61-66.

[Moy94a] J. Moy, "Multicast Extensions for OSPF," RFC 1584, IETF, March 1994.

[Moy98] J. Moy, "OSPF Version 2," RFC 2328, IETF, April 1998.

[MuG97] S. Murthy and J. J. Garcia-Luna-Aceves, "Loop-Free Internet Routing Using Hierarchical Routing Trees," *Proceedings of the IEEE INFOCOM'97*, April 1997, Kobe, Japan.

[Pus99] T. Pusateri, "Distance Vector Multicast Routing Protocol," Internet-Draft, Expires August 31 1999, draft-ietf-idmr-dvmrp-v3-08.

[ShG97] C. Shields and J. J. Garcia-Luna-Aceves, "The Ordered Core based Tree Protocol," *Proceedings of IEEE INFOCOM'97*, Kobe, Japan, April 1997.

[ShG98] C. Shields and J. J. Garcia-Luna-Aceves, "The HIP Protocol for Hierarchical Multicast Routing," *Proceedings of the Seventeenth Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC98)*, Puerto Vallarta, Mexico, June 28 - July 2, 1998.

[Sta98] W. Stallings, *High-Speed Networks: TCP/IP and ATM Design Principles*, Prentice Hall Inc., NJ, USA, 1998.

[Sta99] Stardust.com, *White Paper-QoS Protocols & Architecture*, QoS Forum, July 1999, <www.qosforum.com>.

[ThD95] A. Thyagarajan, and S. Deering, "Hierarchical Distance-Vector Multicast Routing for the MBone," *Proceedings of the ACM SIGCOMM'95*, Cambridge, September 1995, pp. 60-66.

[Win87] P. Winter, "Steiner Problems in Networks: A Survey," *IEEE Networks*, Vol. 17, No. 2, 1987, pp.129-167.

[ZhD93] L. Zhang, S. Deering, D. Estrin, S. Shenkar, and D. Zappala, "RSVP: A New Resource ReSerVation Protocol," *IEEE Networks*, September 1993, pp. 8-18.