QoS Multicast
Using Single Metric Unicast Routing

Kenneth Gustav Enrique Fernando Carlberg

A dissertation submitted in partial fulfillment

Of the requirements for the degree of

Doctor of Philosophy

Of the

University of London

Department of Computer Science
University College London

October, 1999

Abstract

Multicast routing protocols are designed to construct and maintain trees. These trees distribute data on a one-to-many basis in which the replication and distribution of data is accomplished by nodes of the tree. Initial efforts in designing the first IP multicast routing protocol focused on the development of algorithms that produced a separate tree for each source of the multicast group. For those edges of the tree that do not have group members, special control messages, referred to as Prunes, are sent hop-by-hop towards the source to ensure that data is only sent to those nodes that have downstream receivers. However, even with the inclusion of Prune messages, each router in the network was required to retain source/group state for every active source of every group.

Shared tree algorithms and protocols were developed in an attempt to reduce the amount of state stored by source tree protocols. The fundamental design in this approach centered on leaf routers sending an explicit unicast join message to the shared root of the tree, thus grafting a branch from the receiver onto the shared tree. Two of the more widely known attempts to achieve this are known as Core Based Trees (CBT) and Protocol Independent Multicast (PIM). In general terms, the former constructs bi-directional trees in which any on-tree node can be used to initiate the distribution of data to the group members. The latter constructs unidirectional trees were only the root is allowed to initiate the distribution of data to the group members. Both protocols were designed to operate independent of any underlying unicast routing protocol, and both rely on an a priori selection of the shared core.

This approach in building shared trees works well in an environment that is comprised of just best effort service and routing protocols that calculate a single shortest path to a destination. However, the introduction of new service models in the network layer fosters a need to change today's multicast framework so that branches can reflect the desired Quality of Service (QoS) of an application.

My thesis presents a new approach for building shared trees so that branches reflects the desired QoS of receivers. The approach follows a design parameter of CBT and PIM in that the construction of a shared tree is accomplished independently over any unicast routing protocol. However, I introduce a paradigm shift by having leaf routers use a one-to-many join mechanism

to discover multiple paths from which to graft a branch onto the shared tree. These paths can be associated with a variety of metrics or criterion. This allows the network to support applications or system administrators in constructing QoS sensitive trees. Congruent to this selection, is that the shared root is dynamically selected as a function of group membership, which I will show can significantly lessen the impact of core placement in terms of delay from source to receivers.

The innovative approach I use to construct QoS sensitive shared trees is based on a one-to-many join mechanism and is described in my thesis through the Yet Another Multicast (YAM) protocol. The algorithm that acts as the foundation of the YAM protocol represents a variation of the Greedy algorithm and allows the protocol to operate in a scaleable manner in constructing multi-metric trees. I present some of my experiences in implementing the YAM protocol to articulate the problem space of introducing a QoS sensitive multicast protocol within the current IP multicast service model as it exists today.

The primary basis used to validate my claims is done through simulation, and statistical analysis of the data produced from simulated scenarios. Real-time measurements of the design are not presented because an in-depth examination would require an actual test network of considerable depth (i.e., many nodes spanning the boundaries of a network). I describe simulations of the YAM protocol in order to quantify cases in which QoS multicast can be more beneficial than the current single shortest-hop designs being developed today. These simulations also allow me to present a means of representing the impact on the system in using a one-to-many joining mechanism and in having group members migrate over time. Finally, while several existing algorithms are used in tandem to produce QoS sensitive trees, I do not introduce new algorithms in the design, and thus I do not present an in-depth algorithmic analysis in my thesis.

Acknowledgements

I'd like to start by thanking Dr. Gerry Anderson and Vickie Hirt of SAIC for their unqualified support during my adventure at UCL.

My time at UCL was always very enjoyable and relaxing. It was accentuated with the friends I made there, who always had time for a good laugh and a nice conversation. While there are a number of friends in London/UCL that I'd like to acknowledge, I specifically want to mention Lorenzo Vicisano, Jungwon Kim, Paul White, Ian Brown, and especially Nadia Kausar. Nadia is a mate and a spark plug that kept all of us on our toes and in good spirits with her truly independent style. I'd also like to thank Phil Shivers and Selby Hall, old and kind friends from my childhood neighborhood of Bolton Hill in Baltimore (Bawlmer) MD, USA. And finally from the London crowd, Tony Ballardie who brought up the idea of going to UCL in the first place, and Professor Jon Crowcroft – a perfect fit as my advisor. Besides being a friend and a character, Jon has an uncanny ability to exchange views and ideas with seemingly no influence from his own ego. As Lorenzo and I agreed one day over a pint, when talking with Jon, he never met an idea he didn't like, he simply had different levels of interest.

Back in the states, I'd like to thank my friends Padma Krishnaswamy and the old gang: Robert Zarbin & Beth Davis, David Dombrowski & Kathleen O'Keefe, and Maribeth Diemer, and my cousin Laurie Mock.

And then there is the family in Chile, all of whom are wonderful and dear to me. And while its not fair to single out anyone in particular, I'd like to thank my cousins Marcela y Pamela Mendez Koch for being nice pen pals.

Finally, I want to thank my parents Norman and Juanita Carlberg. Their love and understanding holds no bounds, and while words cannot express my love for them, I have been truly blessed to have them as parents and friends.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Multicast routing involves the construction and maintenance of trees used to distribute data to the members of a group. Since its introduction into the IP community by Deering and Cheriton in the form of trees rooted at the source of group traffic [26], the evolution of multicast routing has focused on refinements and new algorithms used to conserve state and minimize overhead costs incurred on the network. Initially, these efforts came in the form of explicit prune messages that truncate portions of the tree rooted at the source of a group. Later, new algorithms, and several iterations of protocol designs were developed to produce a single shared tree for all the sources of the group [5, 27, 66, 73, 88].

One constant in all of these efforts is the reliance on a single unicast metric to build a tree. Often, this metric is based on the shortest hop count between two nodes in a network. Within the context of Best Effort service, single metric routing was sufficient in accomplishing the requirements of most applications used over packet switching networks. This is because Best Effort service does not rely on setting specific bounds on the characteristics of data packets as they traverse a network. Rather, all datagrams use whatever resources of the path that are currently available.

Recently, there has been a concerted effort within the Internet Engineering Task Force (IETF) to go beyond Best Effort service and introduce new classes of services – i.e., Better-Than-Best-Effort service. The initial effort was labeled Integrated Services, which focused on specific bounds on bandwidth and/or delays concerning end-to-end flows [11]. To facilitate this new service, the IETF defined a new signaling mechanism for IP [78] called the Resource Reservation Protocol (RSVP) [110,111]. This protocol was designed to act as a conduit for receiver-initiated reservations that allocated resources on a hop-by-hop basis towards the source of traffic. Reservations could be applied to unicast traffic, or they could be applied to multicast traffic, whereby reservations of downstream receivers would be merged by on-tree parent nodes. Path messages are used by RSVP and sent by the source to indicate how much resource, and what type, should be reserved along a unicast path or branch on the tree. One concern about RSVP and

Integrated Services as a mechanism that operates over the Internet, has been the granularity of the reservation and its end-to-end signaling nature. This concern has constrained this initial attempt at better-than-best-effort service to one that is deployed only in an *intra*-domain or enterprise network environment.

The next evolutionary step in going beyond Best Effort service has been labeled Differentiated Services, which although still being debated, attempts to take a more abstract view of resource reservation and datagram service [58]. Specifically, edges of a network use profiles to mark or drop packets from an aggregated set of sources. Corresponding to this, the application and realization of differentiate services has been constrained to just the ingress and egress points of a domain and its immediate neighbors, as opposed to the end-to-end characteristic of Integrated Services.

An interesting aspect related to adding new services to the Internet is that this has the effect of building a demand to produce alternative routes that are sensitive to a particular Quality of Service (QoS). By this, I mean that the quality of the route must reflect the specific characteristics of the class of service that is being requested. As an example, an application may request that the class of service be based on strict bounds on bandwidth, thereby requesting routes based on available bandwidth. Hence, if the traditional shortest path route does not satisfy the desired class or quality of service, then a new path must be generated. In the context of routing, obtaining a different path for a variety of metrics can be problematic, and generally considered an intractable problem whose solution space is non-polynomial [1, 44]. This problem applies to multicast routing as well, which some consider to be simply a special case of unicast routing. And here lies the crux of the problem, for while the introduction of new services builds a demand for some measure of alternate paths that support different QoS, existing unicast and multicast routing are based almost exclusively on constructing paths/branches based on a single unicast routing metric based on a shortest hop-count between two nodes.

My thesis presents a new approach for building shared trees so that branches of the tree reflect the desired type of QoS for receivers. The approach follows a design parameter of protocols like CBT and PIM in that the construction of a shared tree is accomplished independently over any unicast routing protocol. However, I introduce a paradigm shift by having leaf routers use a one-to-many join mechanism to discover multiple paths from which to graft a branch. These paths,

each reflecting a unicast hop-by-hop response by on-tree nodes to the one-to-many join, can be associated with a variety of metrics or criterion. This allows the network to support QoS capable applications. System administrators, that wish, can build a tree that adheres to one of several types of metrics or criterion. Congruent to the selection of different metrics is the fact that the shared root is dynamically selected as a function of group membership, which I will show can significantly lessen the impact that core placement can have in terms of delay from source to receivers.

The innovative approach I use to construct QoS sensitive shared trees is based on a one-to-many join mechanism and is explored in my thesis through the Yet Another Multicast (YAM) protocol. The algorithm that acts as the foundation of the YAM protocol represents a variation of the Greedy algorithm and allows the protocol to operate in a scaleable in constructing trees of different metrics.

I present some of my experiences in implementing the YAM protocol to articulate the problem space of introducing a QoS sensitive multicast protocol within the IP multicast service model as it exists today. Real-time measurements of the YAM implementation are not presented in my thesis because an in-depth examination would require an actual test network of considerable depth (i.e., many nodes spanning the boundaries of a network) in order to obtain an appreciable examination of the protocol and its design.

The primary basis of validating my QoS Multicast design, and claims related to it, is done through simulation and statistical analysis of the data produced from simulated scenarios. I use simulations of the YAM protocol as a means to quantify cases in which QoS multicast can be more beneficial than the current single shortest-hop designs being developed today. These simulations also allow me to present the impact on the system regarding claims that I make regarding the use of a one-to-many joining mechanism to help construct a QoS tree. Other aspects, such as 'what if' scenarios, are presented in my thesis in order to examine worst case scenarios of the YAM protocol as well as other protocols like CBT or BGMP [5,34]. In addition, I also look at the impact that dynamic or migratory behavior[1] of group members can have in the maintenance of the shared tree.

[1] By 'migratory', I refer to the case in which receivers join and leave the group from different topological locations, as opposed to nodes physically moving through the network.

Finally, while several existing algorithms are used in tandem by the YAM protocol to produce QoS sensitive trees, I do not introduce new algorithms in my design, and thus I do not present an exhaustive algorithmic analysis in my thesis. Rather, some algebraic analysis is used to articulate the anticipated impact that my design may have on a network.

1.1 Thesis Overview

Chapter 2 provides background information describing IP multicast. It discusses the existing Deering-Cheriton multicast model and shows the evolution of initial protocol designs and implementations used to construct multicast trees over an IP (inter)network. From these initial designs focusing on the construction of source-based trees, I present recent work involving shared trees and their ability to conserve state by using a receiver-driven explicit unicast join mechanism used to graft branches on the tree. I also introduce a new term, hybrid trees, to discuss and identify protocols that build both shared trees and source trees for the same multicast group. One aspect that is constant in all of these past and current efforts in multicast routing is the construction of a tree based on a single metric (typically, a shortest hop metric used to identify a path from a receiver to the root). Finally, I present some of the characteristics of multicast applications – the sources and sinks of multicast data. Insight into these applications allows us to understand some of the requirements expected of a tree construction design. Note that the first part of the chapter is presented from a general perspective. Primary topics and sub-categories are presented in more detail as the chapter progresses.

Chapter 3 discusses a new area of multicast routing that goes beyond the efforts discussed in the previous chapter: Quality of Service (QoS) multicast routing. This new area is a reflection of changes in the IP service model and its expansion into better-than-Best-Effort service for unicast traffic. Correspondingly, I present an innovative approach to constructing and maintaining IP multicast trees that are sensitive to a particular QoS (i.e., a metric or criteria other than shortest hop). In this chapter, I present my design for QoS multicast titled Yet Another Multicast (YAM) protocol. This design focuses on building a shared tree based on one of several metrics using a one-to-many join mechanism. I also discuss some related work: one design that places a strong reliance on multi-metric intra-domain unicast routing, and another design, titled QoSMIC, that is an evolutionary effort based on a preliminary design of YAM and postdates the one presented in this chapter.

In Chapter 4, I present a number of issues related to QoS multicast. One of the key points that needs to be addressed in designing a QoS sensitive multicast protocol is its operation over a broadcast LAN and today's current practice of arbitrarily selecting a Designated Router. Another issue that I discuss involves different strategies or approaches by which one can reduce the cost in overhead traffic generated by a one-to-many join mechanism. This latter aspect specifically relates to protocols like YAM or QoSMIC.

Chapter 5 presents an examination of the YAM protocol through simulation. I developed models representing the YAM protocol for the VINT/LBNL/ISI Network Simulator (NS)[2] and tested these models in a variety of ways. These tests are meant to provide a comparison of the trees constructed with different metrics (desired QoS), but with the same topological connectivity and the same receiver set. In this way, I isolated the comparison to just changes in types of metrics.

I use end-to-end delay of traffic through the tree as a way of determining how 'good' a tree constructed with one metric is over one constructed with a different metric. The delay measurements were derived from two perspectives: a) many-to-many communication, involving the average delay of all receivers sending to the group, and b) one-to-many communication, in which the average was derived from one sender (i.e., the core) to the group. Delay measurements, in the form of worst average delay, are used to determine how consistently 'bad' a tree can be for one metric in comparison to another.

I also used the amount of state stored by the network for each type of tree as a means of evaluating the quality of a tree. By this I mean that I used the number of nodes needed to comprise a tree for each metric as a measure of the trees quality – the least number of nodes potentially being viewed as a positive attribute.

In Chapter 6, I continue the examination of YAM with NS and consider the case in which group membership changes over time. This situation allows one to examine the effect that group dynamics have on QoS sensitive multicast trees. I use the dynamics of an actual multicast transmission as the basis for changing group membership in our simulated environment. Beyond this actual case study, I examine a hypothetical scenario representing a type of worst-case

[2] Information on NS simulator can be found at <http://www-mash.cs.berkeley.edu/ns>

scenario for shared tree designs -- the case in which a densely populated set of receivers moves[3] en mass from one topological location to another.

Finally, in Chapter 7, I present a series of conclusions concerning my contributions and the advancements I have progressed in the area of multicasting. I discuss the merits of our innovative design in building QoS sensitive trees. I also present a series of future areas that can be explored and facilitated by my work.

[3] My definition of 'movement' involves receivers joining at one location, leaving, and then other receivers joining at a different location. I do not refer to the case in which receivers physically move through the topology. This latter example is outside the scope of my thesis.

Chapter 2

Background

Multicast involves the distribution of data from source(s) to the members of a group, with the group consisting of one or more receivers. For the sake of simplicity, I equate a receiver with a host, though in reality a host can have a multitude of receivers -- each receiver obtaining data sent to a specific group.

One of the strengths of multicast is that the network accomplishes the replication of data as it is being sent from a source to the different receivers. This is in contrast to unicast replication (e.g., existing push technologies) wherein the source application replicates the data for each unicast destination. In the case of multicast, the source sends a single datagram and certain nodes in the network replicate the data for all receivers. This design feature has the added benefit of reducing the amount of bandwidth used for one-to-many communications. In contrast to broadcast, multicast only involves members of a group, while the former sends data to all nodes regardless of their ' interest' to receive the data.

Trees are used to replicate data sent on a one-to-many basis. By definition, a tree represents a non-cyclical connected graph in which a parent node is connected to a set of one or more children. Each child is also viewed as a parent by the children connected to it. I refer to these children as downstream of the parent, and the parent as upstream from its children. The hop-by-hop collection of downstream children from any parent represents a branch on the tree, And each tree has a single root representing the beginning of the distribution structure.

The current model of IP multicast, known as the Deering-Cheriton model [26], is based on a receiver oriented architecture. By this I mean that the basis for group membership and the corresponding construction or deconstruction of the tree (and/or parts thereof) is based on the actions of individual receivers. Hence, sources never enumerate the receivers of a group and only send data to a single group address. This receiver-driven approach allows multicast or one-to-many communication to potentially scale to vast numbers of receivers. I say ' potentially' because

the algorithms and/or the corresponding implementation used to construct trees can play an influential role in accomplishing multicast over an Internet with millions of hosts. This statement is discussed in further detail below in section 2.1.

Another key aspect of the current IP multicast model is that it involves the unreliable delivery of datagrams on an end-to-end basis. The need for unreliable delivery stems from the necessity to avoid an explosion of positive feedback (e.g., acknowledgements or explicit flow control) from all the receivers to the source of data. Given that unreliable communication is used, the User Datagram Protocol (UDP) is generally accepted as the principle underlying transport protocol for multicast capable applications [77]. However, UDP by itself does not provide enough information to real-time applications requiring time sensitive and sequential reception of datagrams. Examples are audio/video applications that need time stamps and sequence numbers to properly reconstruct data sent by the source. To provide this additional information, the Real Time Protocol (RTP) was defined as a shim layer protocol between the application and UDP. In addition, the Real Time Control Protocol (RTCP) was defined to provide out-of-band feedback information to all the members of the group [84].

A third characteristic of the IP multicast model involves the support of non-member senders. This involves the ability to distribute multicast data from hosts that are not members of the group. In the case of data driven multicast routing protocols like the Distance Vector Multicast Routing Protocol (DVMRP) [80], this is non-issue in terms of its implementation. However, other types of routing protocols that use explicit-joining mechanisms, such as Core Based Tree (CBT) protocol [5], need to add special features to support non-member senders. These protocols and the algorithms they are based on will be discussed later in this chapter.

A fourth element of the IP multicast model involves the separation of control signaling between that of host-to-router and router-to-router. Within the context of the Internet Engineering Task Force (IETF) suite of protocols, the Internet Group Management Protocol (IGMP) is used to support host-to-router communication [25]. Specifically, hosts use this protocol to indicate that a receiver (application) is joining or leaving a group. This action then triggers a subsequent grafting or pruning of branches on a tree by the multicast routing protocol.

I note that there is extensive on-going research work in the area of reliable multicast that can involve explicit feedback which triggers the retransmission of datagrams to some subset of

receivers[4]. The focus of this work, as in the case of the Pretty Good Multicast (PGM) protocol [40], incorporates measures of message suppression. However, this body of research is accomplished above the IP network layer and is thus outside the scope of my thesis[5].

Figure 1 shows the four rudimentary aspects that comprise a multicast tree:

1. Root: The parent node from which the tree emanates.
2. Branch: A segment of the tree that connects a parent node with a child node. The segment can be comprised of a single link or arc connecting two nodes. It can also denote a series of links between a parent and a single child.
3. Receiver: The host node that "*joins*" a group and receives data sent to that group through the tree. This entity also sends "*leave*" messages which terminates the reception of subsequent data for that group.
4. Leaf Router: The node or router at the edge of the tree that connects a receiver to the tree. In certain cases, this node either initiates the pruning of a tree or its construction.

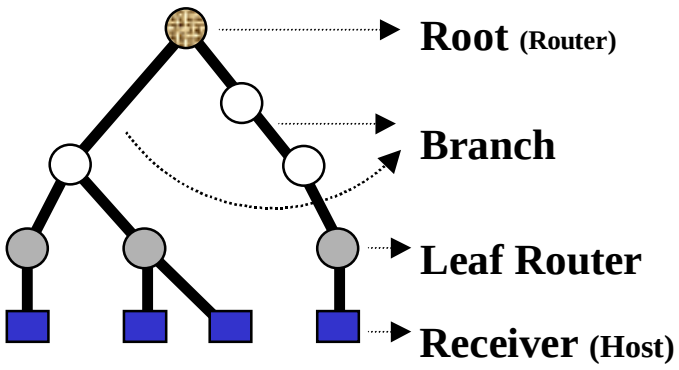


Figure 1: Example Tree

Currently, there are three types of IP multicast routing protocols responsible for constructing the following three different types of trees: Source-tree, Shared-tree, and Hybrids. Most of the protocols discussed below, except BGMP and OCBT, are viewed by the IETF as multicast interior gateway protocols (M-IGP) and are expected to operate within a routing domain. BGMP,

[4] In addition, there has been research involving the use of transport level Forward Error Correction (FEC) to eliminate the need of any feedback and still provide a measure of reliable delivery of multicast traffic [82].
[5] Additional information on the area of reliable multicast and RTP/RTCP can be found in [43, 85, 89]

with its extensions to BGP, is viewed as an exterior multicast protocol and operates amongst domains. The reason for this distinction is primarily associated with the ability of administrators to apply inter-domain policies in constructing trees with BGMP. It can be argued that this distinction is orthogonal to the scaling properties of multicast protocols and is therefore a weak argument for such a distinction.

2.1 Source Trees

Source tree protocols are based on a data driven schema that involves broadcast, reverse path distribution, and explicit pruning to build a delivery tree and ensure that data is only forwarded on interfaces that have downstream group members [64]. In the case of multicast, broadcast has more of a local quality to the extent that data is sent to each interface that does not have exclusionary state for that group -- i.e., state that indicates multicast data for a specific group is not to be sent out a specific interface. This exclusionary state is typically referred to as prune state.

Originally, the distribution method was based on Reverse Path Forwarding (RPF) and was used as a means of preventing loops as information is being broadcast through the network [24, 105]. This was accomplished by only allowing routers to forward traffic through its other interfaces if the input interface used to receive the multicast data is also the output interface when the source address is a unicast destination. This action accomplished two things: it formed a spanning tree from the source to all possible receivers, and it removed the need to retain state per data packet (e.g., sequence numbers) in order to prevent loops. More specifically, the root of the spanning tree is associated with the first-hop router attached to the source-host generating the multicast data for the group. From the root there emanates a series of branches from which other branches emanate, thus forming a source tree.

Subsequent prune information, in the form of unicast control messages, is used to stop the flow of multicast data down interfaces that have no downstream receivers. These control messages are initially generated by stub routers[6] whose interfaces have no hosts that are members of the

[6] A stub router is one that has no downstream neighbor routers.

specific group. Thus, the initial spanning tree used to reach all possible hosts evolves into a source-based tree with the minimal set of branches used to reach all group members[7].

In order to distinguish one source tree from another, routers must retain <source, group> state for each tree that is constructed. This state is either in the form of forwarding state indicating that the node is a member of the tree, or prune state indicating that the node is a not a member of the tree and should not receive nor forward multicast traffic for the specific group.

Given that state must be retained for each source sending data to a given group, it has been widely accepted that source-tree protocols are best suited for dense topologies in which the receiver sets are (relatively and) topologically close to each other. Prime candidates or examples of such topologies would be enterprise networks that are stubs or standalone (intra)networks that are isolated from the Internet.

One of the benefits of source based trees is that their branches exhibit minimal delay between the source and the receivers of the group. In addition, since signaling, in the form of prune information, is a reactive element in tree construction, the initial distribution of data from a new source incurs no delay from tree construction because the data is broadcast to all potential receivers. Examples of applications that would acutely benefit from minimal delay include interactive communication (e.g., audio/video conferences), resource discovery, and time sensitive applications (e.g., stock exchange updates). Note that I acknowledge the fact that all applications can benefit from minimal delay. However, tradeoffs between state and delay have led to variations in multicast routing which is discussed in more detail in subsections 2.2 and 2.3.

The downside of data driven source based trees involves the amount of state that needs to be maintained and the periodic distribution of data to areas that have no receivers. These retransmissions occur due to prune state having timed out, leading to a waste of resources as unwanted multicast traffic is propagated to downstream nodes. Beyond this, there exists the problem of source trees in the presence of distribution boundaries based on administratively scoped addresses or Time To Live (TTL) values placed on data packets. This problem arises when the defined administrative boundary does not span a topology large enough to include all

[7] It can be argued that the source-tree is still a spanning tree for a specific set of nodes, but we make this distinction so that the reader understands the context in which the tree is used (all hosts vs. all receivers of a specific group).

the receivers of a group. Thus, group members receive data only from topologically close sources.

In Figures 2a, 2b, and 2c, I show a three stage process in which a source tree is formed. Figure 2a shows a subset of hosts joining the group via IGMP-join messages. These join messages, as well as corresponding leave messages, can be sent at any time and are orthogonal in relation to sources sending data to the group. For the purposes of clarity, the following figures present a case in which receivers have the joined the group before any sources have sent data.

After these receivers have joined the group, Figure 2b shows which routers are considered leaves. The figure also shows initial data being sent by the source and initially broadcast by the root via RPF. The bold lines in Figures 2b, and 2c, denote those links that have state for the group.

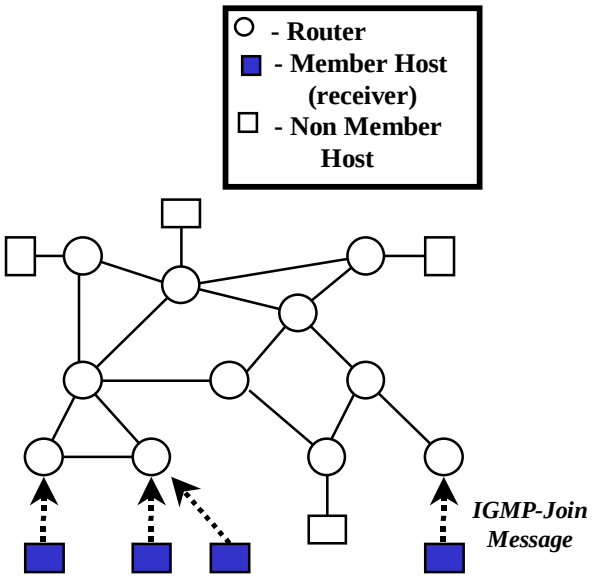


Figure 2a: IGMP-Join

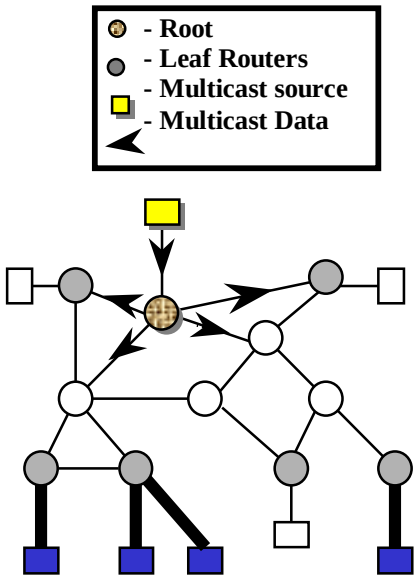


Figure 2b: First Message Sent to the Group

After the initial multicast data has been distributed throughout the network, one is shown in figure 2c subsequent prune messages that truncate the original tree so that data is only forwarded down those interfaces that have downstream receivers. The messages are propagated towards the root and terminated by nodes that have either downstream children or receivers. Finally, one ends up with a source tree whose root is attached to the source of the group.

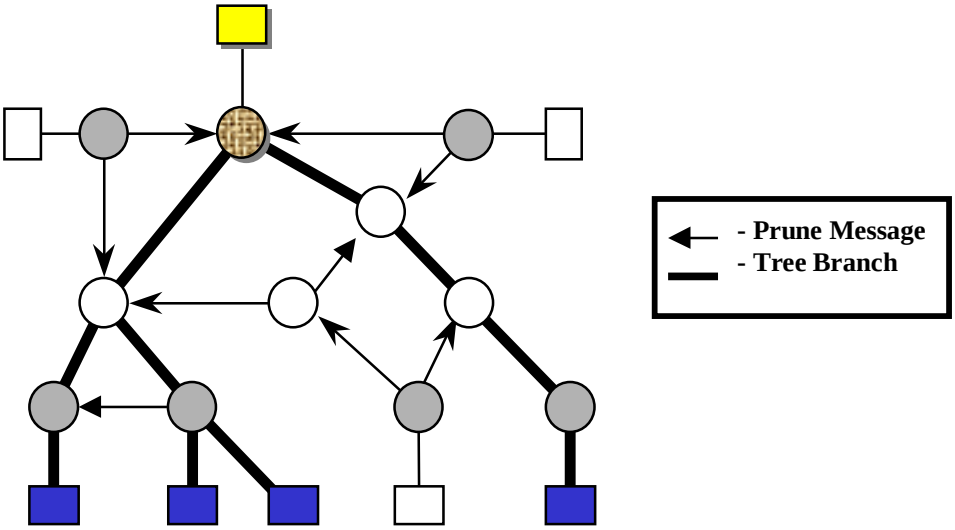


Figure 2c: Prune messages

2.1.1 DVMRP

The Distance Vector Multicast Routing Protocol (DVMRP) was the first IP multicast protocol. It followed the Deering-Cheriton multicast model and was a refinement and augmentation of Dalal and Metcalfe' s Reverse Path Forwarding algorithm [24, 98]. DVMRP is essentially a derivation of the Routing Information Protocol (RIP) which exchanged reachability information of subnets based on distance-vector routing [54]. As a result, routers that support both unicast and multicast build and maintain two corresponding sets of routing tables. The protocol also forms a single flat hierarchy that computes routes and initiates prunes based on the multicast routing information distributed by neighbor routers.

While the protocol was initially targeted towards small dense topologies, it became the basis for the Multicast Backbone (MBone). The MBone is a virtual overlay of the Internet that supports multicast routing and spans over 8,000 networks [37, 63]. The connectivity of the MBone is primarily accomplished by encapsulation tunnels that connect multicast capable routers.

Over the years since its initial introduction, modifications have been made to DVMRP to cope with the relatively large and ever increasing set of networks that comprise the MBone. One such modification involves the use of source masks to help aggregate the number of sources per group [80]. The manner in which the aggregation is accomplished in the form of a variable sized mask. However, since source trees have roots at each source, the size of mask can only significantly

increase towards the edges of the trees for a specific group. This means that the middle of the tree has to maintain a significant amount of state for each source. Additional information on sources and the expected behavior of certain applications is discussed in section 2.5.

DVMRP forwards initial data on a truncated broadcast tree, which then gets pruned to form the multicast tree. The truncation is accomplished by its routing protocol which forms child parent relationships before any data is sent. State is maintained for all potential sources and all prunes that have been received.

2.1.2 PIM Dense Mode

Protocol Independent Multicast is a tree construction protocol that operates independently over any unicast routing protocol. Currently, PIM is divided into two perspectives: dense mode (PIM-dm) and sparse mode (PIM-sm) [32, 33]. The former refers to topologies of receivers that are relatively dense, and thus constructs source-based trees. The latter focuses on receiver sets that are considered sparse and/or topologically distant from each other -- the objective being to build shared trees and minimize the amount of state stored in the network. Below I discuss the dense mode version of PIM, while the sparse mode is presented in section 2.3.2.

PIM-dm is very similar to DVMRP. It is a data driven multicast routing protocol that constructs source based trees rooted at leaf routers attached to a source. Data is initially forwarded via reverse path broadcast (i.e., a broadcast tree). Subsequent prune control messages are used to form the multicast tree. Since PIM operates over any unicast routing protocol, it does not distribute or maintain a separate routing table of its own. Instead, it uses the routing table constructed by the underlying unicast routing protocol to determine if ingress multicast traffic conforms to RPF checks. PIM does build and maintain a separate Forwarding Information Base (FIB) to store the output interfaces used to forward incoming multicast data. PIM-dm is also a softstate protocol that uses data messages to refresh its state.

2.1.3 Multicast Open Shortest Path First

OSPF is a link state intra-domain routing protocol that operates at two levels: 1) within an Area (an arbitrary collection of subnets), and 2) among Areas of a domain. Each router floods Link State Advertisements (LSAs) to the other routers that are peers at the same level. From these advertisements, routers are able to construct their own routing tables using a Dijkstra algorithm [69]. MOSPF specifies extensions that allow LSA messages to contain information regarding routers that have neighbor hosts which are members of a multicast group [70]. When a new receiver joins a group in a subnet, and no other receivers exist, then the designated router advertises this information to its peers via flooding.

MOSPF builds source-based trees. When a new source sends its initial data to the group, downstream routers construct a shortest path tree from the perspective of each downstream node to the receivers (actually, subnets) of the group. This calculation identifies the downstream nodes and subsequently forwards data to those neighbor routers. The source address is used to distinguish one tree from another tree for the group.

Given that MOSPF constructs source-based trees, it suffers from the same problem of state explosion in cases where there is more than one source for a group. In addition, other routers that are not members of the tree still need to maintain state via the LSAs that contain multicast group information.

2.2 Shared Trees

Given the concern about the potential explosion of state stored by all routers for source-based trees, shared tree designs were developed to help conserve state as well as reduce the cost to the network in signaling overhead.

The concept behind this effort involves the use of a single tree for all sources of the group. Specifically, a shared root or core is used as a point of reference from which to graft branches onto the tree. The algorithm used as the foundation behind this approach is attributed to Wall and his work on the construction of a single tree for multi-destination delivery of packets [103]. In Wall' s thesis, he stipulates that one cannot expect the single shared tree to achieve minimal delay or cost for all leaves in comparison to trees rooted at each source. However, "we may be able to

do fairly well, and the simplicity of the scheme may well make up for the fact that it is no longer optimal" in terms of delay or cost from any leaf to all other leaves of the tree [103]. Wall also proves that the maximum delay bound of a core based tree is twice that of a shortest path tree.

From this algorithm, additional designs have been put forth to address the need for dynamic group membership and tree maintenance due to connectivity failure. In addition, the orientation of tree construction of these new designs has been shifted from a source perspective to a receiver initiated perspective. In other words, recent work in the area of shared tree construction has focused on receiver-initiated instantiation of branches, as opposed to the sender explicitly enumerating the leaves the shared tree. This aspect produces a more scalable design in cases where the receiver set for a group is large[8].

In using a receiver initiated approach, shared trees are built with an explicit joining mechanism. Within the context of today' s protocols, explicit unicast join messages are triggered by IGMP-join messages and are sent from the leaf router to the shared core. This action results in the installation of state, i.e., grafting a branch, along the path of the join message. Depending on the protocol and implementation design, this state can be ' soft' , requiring timers and refresh control messages. The state can also be ' hard' implying that an explicit message is needed to finalize the installation (or removal) of state for the group. More specific information concerning these approaches are presented below in the following subsections.

In contrast to the broadcast and prune schema of data driven multicasting, the unicast join mechanism of today's shared tree protocols conserves system overhead by minimizing the number of nodes that receive control messages (signaling). This means that other off-tree nodes do not need to keep prune or exclusionary state information. Thus, only on-tree nodes retain group information -- commonly denoted as **<*,g>**, where ***** is a wild card for all sources, and **g** represents the group.

Figure 3 presents a generic representation of the construction of a shared tree. Barring protocol specifics, most of today's shared tree protocols follow this generic model. In conforming with the receiver-driven model of IP multicast, the explicit join is triggered by an IGMP-join message sent by the receiver/host to the leaf router, which installs state for the group in the leaf router.

[8] By today' s standards, from thousands to millions of receivers.

This leaf router then sends the explicit join on a hop-by-hop basis along a path to the shared core -- installing state along each node of the path.

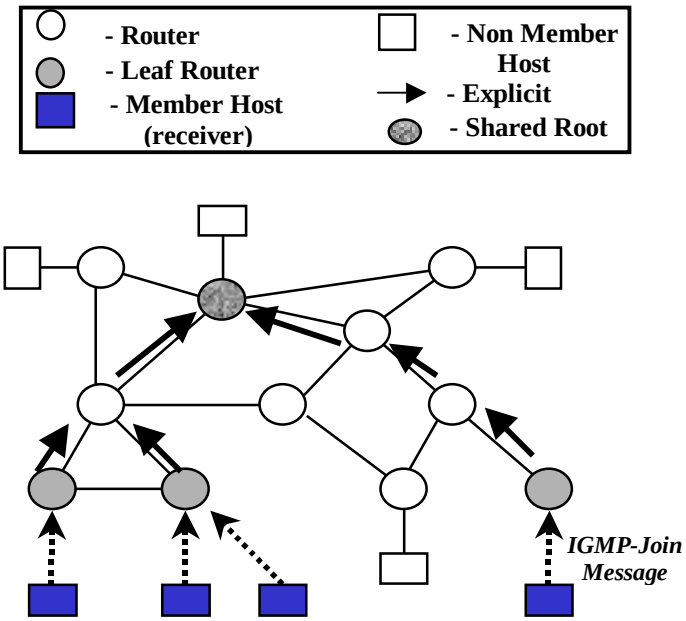


Figure 3: Generic Example of Shared Tree Construction

Core Based Tree (CBT) and PIM sparse mode, denoted as PIM-sm, are two existing protocols used to build shared trees. While there are several distinct differences between the two, they share a common feature of operating independently over any unicast routing protocol. It is also implicitly understood that both CBT and PIM construct trees based on the shortest hop metric, since most routing protocols use this single metric to calculate destination reachability. In addition, both protocols install state on a hop-by-hop basis from the direction of the leaf router towards the shared root. This is of particular interest since the ' best' path from the leaf to the root may not be the same from the opposite direction because of existing load from other sources.

Over the years, CBT and PIM have evolved since their initial design. Below I briefly discuss the design of the first version of CBT, and a subsequent design known as Ordered CBT. I classify these designs as ' pure' shared tree protocols because they do not support the inclusion of source

state. In section 2.3, I discuss CBTv3 [7] and PIM-sm and refer to them as Hybrid Tree construction protocols because they can build both source and shared trees for the same multicast group.

2.2.1 CBTv1

This version of CBT [5,6] contained a number of distinctive features representing several design choices that separated it from the subsequent PIM-sm protocol [27]. One of the principle differences involves CBT' s construction of bi-directional trees , which allows it to support any-point distribution. This type of distribution relates to the fact that once a tree is constructed, data can start to be distributed from any point or node on the tree, since it is a non-cyclical collection of paths. Hence, data can not only start to be distributed from the core or shared root, but it can also be initially distributed from any leaf router. The fundamental rule in its distribution is that multicast data is sent out to all the interfaces that have state for the group except the interface that the data was received on. Note that this is similar to the rules stipulated for RPF, but without the added requirement of the input interface being the same one used to send data to the source of the multicast traffic.

Another distinctive design feature of CBTv1 is that it supports the notion of a primary and a set of secondary cores. The primary core equates to the shared root, while secondary cores act as a sequential list of nodes that can be used as the shared root in case the primary fails. This allowed the protocol to support a measure of fault tolerance as well as a rudimentary means of employing traffic engineering in constructing a shared tree.

Finally, CBTv1 was classified as a hardstate protocol that used an ack/nack exchange of messages to install state for a group and thereby grafting a branch onto the tree. A Keep-Alive message was used to check the status between a child and its parent, and vice-versa. If the keepalive and subsequent ack messages were not received, then the link was considered broken and a new branch, reflecting a different path, would be grafted. One could argue that the use of keepalives reclassifies CBT as a softstate protocol, wherein state needs to be periodically refreshed, or else it times out and state is removed. But for the sake of consistency with previous literature, I shall continue to refer to CBT as a hardstate protocol.

2.2.2 Ordered CBT (OCBT)

This protocol is an augmentation to CBTv1 and is designed to address a design deficiency of that protocol as well as support a more robust set of cores for a group. In addition, OCBT incorporates extensions so that it can be used to construct *inter*-domain branches within the architecture of the Hierarchical IP multicast routing scheme protocol (HIP) [60, 88]. HIP can operate over BGP and use its policies in constructing branches on the *inter*-domain tree.

The design deficiency in CBTv1 involved situations where instability in the underlying unicast routing could cause loops during the construction of the tree. Specifically, if the path to the primary core was broken, then a downstream node would attempt to graft a new branch onto a secondary core. If the secondary core was off-tree, it would then attempt to graft a branch onto the shared core. This subsequent grafting process could include an existing downstream node which would be undetected because the 2 types of hop-by-hop grafts (on-tree router to secondary core, and secondary core to primary core) are considered two different and distinct protocol actions by the CBTv1 protocol.

OCBT fixes this problem by applying labels to cores and placing cores at arbitrary levels in a hierarchy. Further, OCBT cores only exchange information with other nodes at its level, or the level directly above or below it. This allows the protocol to constrain the distribution of control information used to construct and maintain trees. This approach also removes the problem of an implosion of explicit joins sent to a single core for the group.

As in the case of CBTv1, core placement is treated as an open issue that can be potentially resolved in a variety of ways. One may involve periodic, though infrequent, flooding of information by randomly chosen nodes which are arbitrarily selected as cores for the tree. Another can involve the use of the Domain Name Service (DNS) and augmenting the information it stores and distributes [70, 71, 97]. Again, this second approach involves a measure of random selection of the core for a group.

2.2.3 Simple Multicast

The Simple multicast protocol is an on-going effort that can be considered a derivative of CBT and PIM [76]. The motivation behind its design is to specify a protocol less complex than CBT or PIM and that:

1. Supports bidirectional trees
2. Obviates the need for a multicast address allocation mechanism
3. Increases the multicast group ID space

The first item relates to any-point distribution and is considered desirable in cases where many-to-many communication occurs within the group. This is because there is no requirement that traffic from all sources be encapsulated by the leaf router and sent to the root for distribution down the tree. The second item removes a dependency on another suite of protocols in order to effectively accomplish *inter*-domain multicast. This dependency not only relates to supporting a third-party functionality but also places a new requirement in the existing IP multicast service model. Finally, the third item increases the multicast address space by concatenating the core IP unicast address with the multicast address to form the multicast identifier. This also provides hooks for supporting traffic engineering and addressing the single point of failure problem by using different unicast core addresses.

To accomplish the above, the Simple protocol proposes a substantial change to the IGMP protocol and the application protocol interface (API) used to join a group. This change involves the application supplying the IP address of the target core, which is then included in the IGMP-join message[9]. The manner in which the application obtains the core address is an open issue, though the authors suggest obtaining the information through one of several well known mechanisms: the Domain Name Service (DNS), Session Directory [51], web advertising, etc.

[9] We view this as a 'substantial' change because of its potential impact on the existing installed base of hosts that use the existing group-only API. However, this viewpoint does not mean that the problem is insurmountable.

2.3 Hybrid Trees

Two primary concerns associated with shared tree protocols are core placement and traffic concentrations. Most of today's shared tree protocols make a priori selections of which node is to be selected as the shared root for a multicast group. This selection is made before any group members join the group and thus is orthogonal to the topological location of receivers. If the selected node/core is close to the receiver set, then the end-to-end delay between source and receivers will most likely be reasonable in comparison to a tree placed at the source of the multicast data. However, the core could be selected at a location far away from the rest of the receiver set, which can adversely impact end-to-end delay of data[10].

The concern regarding traffic concentrations arises in cases where several sources are sending data to the same group. Using just a single shared tree for data distribution, all on-tree nodes are then candidates for experiencing congestion. The severity of the congestion is dependent on the bandwidth of a link, the cumulative traffic from group sources, and other flows transiting the on-tree node.

Hybrid tree protocols attempt to address these concerns by constructing a single shared tree for relatively low amounts of group data, and also construct source trees for high rates of data transmission from a given source. Since hybrid tree protocols can produce a combination of a shared tree and source tree(s) for the same group, it is presumed that some percentage of the source tree branches will have disjoint paths from the shared tree. It can also be presumed that the leaf portions of the shared tree will share the same links as the individual source trees, but segments that are closer to the shared root probably will not have the same amount of state or traffic loads for all sources. Thus, it is presumed that segments of a hybrid shared tree will not be subject to the same amount of load concentrations as a ' pure' shared tree.

To date, no direct examination has been made on alleviating load concentration using hybrid trees on actual multicast networks like the Mbone. On a related note, this raises the issue of what is the tradeoff value in producing hybrid trees. The following protocols produce hybrid trees and use specific criteria to migrate the single shared tree into one that combines a series of source trees to

[10] Simulated results to bolster these claims are presented in Chapter 5 and 6.

achieve: a) lower delay for a specific source, b) provide a means of bridging domains that have source tree protocols with other domains that have shared tree protocols, and/or c) reduce load concentrations on portions of the shared tree.

In the case of PIM-sm, the protocol specifies a heuristic that uses data thresholds to trigger a switch to a shortest path tree for a given source of the group. If the source is sending at a rate above the threshold, the leaf router then produces a source tree from itself to the other receivers. The heuristic is rather simple and easy to implement, but is also orthogonal in terms of conserving state in the network. In addition, the threshold is somewhat arbitrary[11] and unrelated to the characteristics of specific applications. The downside of this type of migration is that it can lead to oscillation in tree construction/deconstruction, corresponding in fluctuations in state storage and signaling overhead. From a more general perspective, the migration is sender initiated and may be contrary to the policies of the receivers in attempting to conserve state. If the number of sources that trigger source trees is low (one or two), then the impact can be considered minimal. But if the number is high, then the conservation of state via the shared tree is negated.

Below, I present a brief description of three types of hybrid protocols that have been, or are being, advanced in the IETF.

2.3.1 CBTv3

The third version of CBT is an evolutionary specification of the protocol that retains most of the design features of version 1 (e.g., hard state and bidirectional trees) [7]. A bootstraping mechanism[12], as proposed for PIM-sm [35], was adapted as a means of distributing <core/group> mappings. Corresponding to this new type of mapping is CBTv3's omission of the list of secondary cores from CBT-join messages. This helped simplify the protocol, but also placed the burden of identifying other nodes as candidate cores onto the bootstraping mechanism.

[11] The default threshold is generally above the amount sent by the Real Time Control Protocol (RTCP), but less than the steady state behavior of the application [112].

[12] Is defined as an algorithmic mapping of group addresses to shared roots or cores. This requires a periodic advertisement of the cores as well as a common hash function used to map any group address with a core address[51].

Source state information was added so that CBT could be used as a transit multicast protocol that integrates domains supporting data driven multicast protocols. In an effort to reduce the amount of source state retained by the transit domain, variable size masks were added so that aggregate information could be stored. It should be noted that the inclusion of **<s,g>** state only applies to the on-tree nodes within the transit domain. Further, it is only initiated by border routers -- routers at the edges of a domain. This approach to adding **<s,g>** state is different than that supported by PIM-sm.

An additional option was added that would allow CBT to graft ' one-way' or unidirectional branches. This could be applied to the non-member sender and avoid continual encapsulation/decapsulation of messages. Other options include the ability to specify exclusionary or inclusionary lists of groups associated with the tree. Presumably, this would allow administrators to enforce a limited set of policies for the nodes comprising the tree.

2.3.2 PIM Sparse Mode

PIM-sm is a softstate multicast protocol that is designed to operate over any unicast routing protocol [32]. Explicit join messages are periodically sent from leaf routers towards the core, which installs state on a hop-by-hop basis along the path. If a ' better' path is available, the join messages are rerouted and new state is installed, while the old state on the old path is timed out and removed.

Unlike the any-point distribution schema of CBT, PIM-sm uses single-point distribution, wherein only the shared root is allowed to initiate the forwarding of data down the shared tree. As mentioned earlier, PIM-sm uses a threshold heuristic to determine if an additional source tree is to be constructed from a source sending data to the group. If the threshold is surpassed, then the source tree is constructed. If the data rate remains below the arbitrary threshold, then the data is encapsulated by the leaf router attached to the source and forwarded to the root, where it is then decapsulated and forwarded down the tree.

PIM-sm, as well as CBTv3, use a single core for any given tree in a network[13]. A few years ago, Handley put forth a proposal called Hierarchical PIM which described a shared tree environment

[13] PIM uses the terminology of Rendezvous Point (RP) instead of Core.

comprised of a hierarchy of Cores [52]. Receivers would graft branches onto the core at the lowest level, which in turn would graft a branch onto the next level. In a sense, each level would represent a larger topological area roughly resembling that used with scoped addresses– e.g., network, domain, region, continent. The selection of a core for a given level was accomplished via a hash function. The approach of having multiple cores for PIM was eventually abandoned in favor of the recent work involving the Multicast Source Discovery Protocol (MSDP) [108] – a design in which shared trees are 'connected' with a series of source trees.

2.3.3 Border Gateway Multicast Protocol (BGMP)

BGMP is a protocol extension to the Border Gateway Protocol (BGP); allowing it to construct *inter*-domain multicast trees [34, 95]. The formation of these trees is a product of the M-IGP within a domain as well as the multicast address allocation suite of protocols being developed under the Multicast-Address Allocation (MALLOC) working group of the IETF [50, 53, 94]. When the M-IGP establishes a branch that spans the domain and reaches a border router, the border router extends the branch on an *inter*-domain basis.

Some of the design principles of BGMP are very similar to that of CBTv3. By default, bidirectional trees are constructed amongst domains. Source specific unicast branches are grafted in order to support domains whose M-IGP is a data driven broadcast type of protocol (e.g., DVMRP).

One difference in design between the two protocols involves BGMP' s treatment of an entire domain as a shared core, as opposed to CBT' s use of a specific unicast address of node. Given the level of abstraction and possible multihomed connectivity, BGMP has the potential to adequately address the single point of failure problem of a shared root architecture.

The use of a multicast address allocation mechanism, and its assignment of addresses to specific domains, addresses the issue of core/group mappings. The MALLOC suite of protocols dynamically assigns blocks of addresses to domains, which in turn can subdivide the blocks into more granular blocks (e.g., an ISP allocating sub-blocks to its customer stub domains). These protocols are also capable of reclaiming blocks for future reassignment. Unfortunately, the allocation process, does not necessarily correlate to an aggregation of group state.

2.4 Single Source Trees

Recently, Holbrook introduced a design for constructing single source trees called ExPlicitly Requested Single-Source multicast (EXPRESS) [56]. In this model, only one node sends data to the group, with the rest of the receiver set acting as pure sinks. If the need for interactive communication arises, then either a new tree is formed for the new source, or a Session Relay node is used to distribute the traffic to the other receivers. The Holbrook-Cheriton model is in contrast to the existing Deering-Cheriton model in two ways: the non-member sender is not supported, and trees are based on a combination of source address (of the source) and the group address, which forms a channel. Channels are advertised by an upper layer mechanism that is out-of-band from the network level tree construction part of the design. The perceived benefits of this approach involve:

1. Moving the group-to-source mapping to the upper layers,
2. Simplifying the multicast address allocation problem, and
3. Preventing unwanted or malicious sources of a multicast group

Applications that could benefit from such a design would include lecture or single transmission types of applications, such as radio or television over the Internet. Given that the Express model is based on a single-source model, it can facilitate the use of access lists to limit which receivers are allowed to join a channel. The EXPRESS design also includes a count system that provides feedback regarding the number of receivers that have joined a channel. This would seem beneficial in realizing a monetary billing system for multicast feeds.

2.5 Characteristics of Multicast Applications

In general, I characterize applications in terms of bandwidth, delay, and loss requirements, as well as the need, or lack thereof, of real-time communication. Additional requirements, and specific bounds help refine but not change my characterization. The relationship to Quality of Service (QoS) is discussed in more detail in Chapter 7.

Within the context of multicast routing, the topological location and number of distributed applications participating in group communication have a direct correlation in the structure of the

tree and its size in relation to the (inter)network as a whole. One can further characterize these applications in terms of the number of sources, and their interactive nature, over the life of a session. Roughly, I identify three basic types:

1) Single-Source
2) Many-to-many
3) Conference

The first case is a single source sending to the group. One can exemplify this in terms of sending a radio or television broadcast over a network. The second case is the opposite of the first and represents a communication model in which all (or most) receivers are acting as sources and are sending data to each other during the life of the session. Examples of this case include interactive applications like multi-player gaming or Distributed Interactive Simulation (DIS). The third case is more amorphous and is meant to represent a balance of the first two in which a few sources are sending data to the group at some point in time.

Multicast applications use the User Datagram Protocol (UDP) as the transport protocol over IP. UDP is stateless and does not provide end-to-end feedback or flow control, as does TCP. This has the benefit of alleviating a potential implosion of information at the source, but it also lacks the ability to respond to congestion in the network. To compensate for this, as well as provide information (e.g., timestamps) for real time communication, most multicast applications use a shim layer known as the Real Time Protocol (RTP). Bundled with this protocol is the Real Time Control Protocol (RTCP), which provides feedback to all members of a group via a different group address. This has the interesting property of characterizing all RTP/RTCP based applications as many-to-many type communication. To address this dichotomy of attributes, PIM-sm uses the threshold mechanism as a way of retaining shared trees for RTCP communication and source trees for application data sources.

2.6 Summary

This chapter has presented an overview of IP multicasting. I have discussed the two primary types of trees that are constructed and maintained by today's protocols: source trees and shared trees. This chapter has also introduced the term hybrid trees. This third type of tree provides a

more concise definition of those protocols that construct an integrated set of both shared trees and sources trees for the same group – the actual type being dependent on data driven heuristics.

I have also discussed some of the characteristics of known multicast applications. These range from the many-to-many communication in which all receivers are also sources of substantial traffic, to the pure lecture type of application (e.g., radio-over-the-Internet) in which there is only one primary source of data to the group.

Chapter 3

Quality of Service Multicast

Introduction

In Chapter 1, I gave a short description on the motivation for alternate paths in helping to construction a shared tree within the context of the existing IP services and routing architecture. This reasoning stems from the fact that the inclusion of services other than Best Effort implies a need to discover alternate paths when the existing default shortest path cannot satisfy the type or quality of service that is requested of the network. In this chapter, I start with a more expanded presentation of what is Quality of Service and how it is viewed from three different perspectives: the application, the logical IP network, and subnetworks like ATM. Following this, I present the topic of QoS IP multicast and a design by which it can be realized over an internet.

3.1 QoS and IP Multicast

Quality of Service is a term that tends to elicit a healthy debate among engineers and researchers of different backgrounds and disciplines. One aspect that can be of agreement is that QoS in a distributed environment implies a quantifiable treatment of data sent from source to receiver(s). How this 'treatment' is defined and how it is supported by the network is the crux of debate. However, one can defuse the debate by compartmentalizing the subject in terms of 2 perspectives: the application and the network. In addition, the network perspective can be subdivided further into 2 schools of thought.

From the perspective of the application operating over a network, one can generalize QoS in terms of its end-to-end characteristics involving packet delay, throughput, and/or packet loss. I say generalize, because applications can have a variety of requirements that can be derivatives of a given quality of a service. For example, the notion of delay can be in terms of a maximum and a minimum bound that can be used to reduce the amount of jitter experienced by a stream of packets. In another example, some applications may place a high priority in high bandwidth, but

have little interest in delay (e.g., file transfer), and vice-versa (e.g., remote login). Hence, a single characteristic, or some combination or derivative thereof, can become aspects that define the QoS of an application. In turn, these aspects are expected to be supported in some fashion by the network. The availability of resources, as well as the capability to discover and reserve them, determine the measure by which a network can provide QoS support to the application.

From the perspective of an IP network, the responsibilities of satisfying the requested QoS of an application are actually expanded and divided into two separate and distinct areas: routing and resource reservation. The first involves the discovery of alternate paths in the event that the default route cannot satisfy the type or quality of a requested end-to-end service. In taking a more definitive perspective, RFC-2386 defines QoS Routing as, *"A routing mechanism under which paths for flows are determined based on some knowledge of resource availability in the network as well as QoS requirements of a flow"* [83]. It is important to note that the definition does not include the responsibility of reserving resources that adhere to the desired level of QoS for the flow. Instead, this latter aspect is realized as a separate signaling mechanism as defined in the RSVP protocol [110]. Further, the qualitative element of the reserved flow, or its specific upper and/or lower bound, is defined by either the Integrated Services model (in the form of Guaranteed Service or Control Load [11]), or by the ongoing effort known as Differentiated Services [106], which involves a more aggregated set of flows. This separation of finding alternate paths and reserving resources was a deliberate design choice by the IETF so that different solutions for each can be inserted and augmented independently.

A different approach to that of the IETF's architectural model of QoS routing can be found in other network architectures. For example, ATM combines the responsibility of alternate path routing and resource reservation into the same mechanism. Through its combination of Private Network-to-Network Interface (PNNI) routing protocol (based on Dijkstra's link state algorithm), and services like Constant Bit Rate (CBR) or Available Bit Rate (ABR), a path or virtual circuit maintains a negotiated service level throughout its lifetime. When there is a need to change the characteristics of the circuit, then the existing one is torn down and new signal messages are exchanged to establish a new circuit. The use of such an integrated mechanism, its reliance on a single link-state routing protocol, and its construction of source-based trees, places this approach outside the scope and direction of my thesis.

In designing an IP QoS multicast routing mechanism, which is presented in the following subsection, I have followed the established IETF paradigm and focused my design on primarily the discovery of alternate paths using a variety of types of criterion for defining a path. This discovery process, as well as the subsequent grafting of branches, does not entail reserving the resources along a new branch by the multicast routing protocol per se. Instead, this responsibility is left to a separate signaling protocols like RSVP.

The issue remains of how to achieve alternate path routing. In Chapter 2, I mention the existence of MOSPF protocol used to construct a multicast tree using an underlying link state protocol. In [83], proposed QoS extensions are added to the link state protocol so that available link resources as well as existing link resource reservations on a router are advertised to other routers. The advertisements are flooded to all the routers within an OSPF area and to all the level-2 routers used to connect areas, upon which the Dijkstra algorithm is used to produce a minimal cost shortest path tree. The use of Explicit Route Advertisements (ERAs) reduces the amount of flooding injected into the system, but a measure of distributed information and subsequent computational resources are still exercised whenever a change occurs in a link's resource (e.g., a new reservation for a new flow or group).

On a related note regarding Dijkstra's algorithm, [113] and [114] present a theoretical perspective that introduces constraints to produce a source-based tree that conforms to a minimal cost that is bounded by an arbitrary maximum delay. The Waters heuristic accomplishes this goal by extending Dijkstra's algorithm to retain alternate paths that exist within the arbitrary delay bound as the tree is constructed. Individual branches are formed from the perspective of the minimal cost of an off-tree node towards the existing tree. The Crawford heuristic is a variation of Waters and instantiates a branch by using the cost of the path from the off-tree node to the source or root of the tree. However, [113] shows that these heuristics, as well as other constrained based designs, can perform better than their counterparts under varying conditions. Thus, it is determined that a hybrid approach, in which different heuristics can be employed in tandem, seems to produce the best results.

While this approach to supporting QoS multicast routing seems to show promising improvements in protocols like QOSPF, it should be noted that this theoretical effort focuses on the requirement of an underlying link state routing algorithm. This requirement runs contrary to one of the design principles of my thesis – the use of <u>any</u> type of underlying unicast routing protocol to construct a

QoS sensitive multicast tree. Thus, the specific work in any underlying routing protocol is not taken into consideration by my QoS IP multicast routing design.

QoS IP Multicast

Having discussed QoS from a broad perspective, how does it relate to IP multicast? QoS IP multicast involves the construction and maintenance of trees that reflect a desired service other than that provided by a default uniform metric or criteria. Most deployed *inter*- and *intra*-domain unicast routing protocols use a single metric based on the shortest numbers of hops between two nodes to calculate routes. The reason for this reliance on a single metric is that it has been shown that the selection of a unicast path using multiple unrelated metrics is a Non-Deterministic Polynomial (NP) Complete solution space [62]. Given that receiver driven shared tree construction involves a series of source-destination (leaf-root) segments, the same postulation holds true in using different metrics for constructing shared multicast trees. However, if one limits the problem space by constraining the set of attempts in finding solutions to a polynomial set, then one has the potential of producing QoS sensitive multicast trees. I say potential because by limiting the number of attempts, one may miss possible solutions.

3.1.1 Related Work in QoS IP Multicast

In this subsection I present two related efforts used to construct QoS multicast trees. One involves the a priori calculation of a limited set of intra-domain routes in order to provide a choice of paths used to construct a tree. The other approach is a follow-on approach to a previous version of my work known as Yet Another Multicast (YAM) QoS protocol. It adds new features to the YAM architecture in order to constrain the cost of the one-to-many join mechanism.

In the subsequent sections, I discuss the goals, architecture, and specification of the YAM protocol and its use of a one-to-many discovery mechanism. I also discuss implementation issues as well as problems that arise when constructing a QoS sensitive tree over broadcast Local Area Networks (LAN). Finally, I discuss techniques used to reduce the control message overhead cost in using a one-to-many join mechanism.

- **Match OR Fail (MORF): Alternate Path Multicast Routing protocol**

MORF can be considered a reaction-type of protocol [108]. Its primary purpose is to find an alternate path to graft a new branch if the default or current branch does not satisfy the requirements of an application. The MORF architecture is divided into two parts: a QoS Manager and a Route Construction Agent (RCA). The QoS manager is a separate entity that can be co-located with the receiver on a host, or it can be attached to a leaf router. It is responsible for informing an on-tree router that the current branch does not meet certain requirements and that an alternate path is needed to graft a new branch.

The RCA is the entity that goes through the process of finding an alternate path. The important aspect to note is that the RCA calculates localized static routes. Hence, if a multicast tree spans several domains, only the alternate path functionality within a domain is exploited. In this way, routing updates do not have to be distributed throughout each domain. Beyond the exploitation of potential localized multipath routing, RCA' s that are scattered throughout the internetwork can be used as a third party proxy for constructing a new path between the receiver and the source. Specifically, if a bottleneck exists between the RCA near the receiver and the source, the RCA can then attempt to find a new path between itself and another RCA that is near the source. Paths between RCA' s are calculated via the Dijkstra algorithm [108].

Explicit routes are used by the RCA to protect against possible loops with alternate routes. If the explicit route reaches an on-tree node whose upstream path does not match the explicit route, a failure notification is sent down and the RCA updates its table to reflect the difference. Otherwise a match notification is sent back to the RCA. This Match or Fail characteristic is the basis for the term MORF.

One of the advantages of MORF is that its design allows it to be adaptive in its application to either a source tree or shared tree protocol. This is because action is only taken if the current or default branch does not meet the requirements of the application. However, the success of MORF is dependent on the existence of either multi-path intra-domain routing protocols, or in a distributed set of Route Construction Agents. In addition, there still exists a need to advertise the presence of other RCAs and to calculate Dijkstra routes on known RCAs. This implies that the set of RCAs should not exceed a number that becomes unmanageable in terms of distributing information and calculating routes.

- **QoSMIC: Quality of Service Multicast Internet protoCol**

QoSMIC is a recent effort that attempts to improve upon an early version of the YAM protocol, which is presented in the next subsection [38]. As in the case of YAM, QoSMIC attempts to build shared trees using a one-to-many join mechanism. This type of join searches for on-tree nodes that send responses to the originator of the join. The originator then uses the response(s) as the basis for deciding which path is to be used to graft a new branch onto the tree.

One of the concerns attributed to using a one-to-many join is the impact it has on the network in terms of the number of control messages that are generated. One strategy to reduce this impact is to use an expanding ring search for nodes on the existing tree, as opposed to a single one-to-many join that is propagated with a large Time To Live (TTL) value. However, the authors of QoSMIC have shown that an expanding ring search can be quite costly (exponentially higher than QoSMIC as the average degree of a router increases linearly). Hence, they introduce a new entity known as the Manager Router (MR). This entity acts as an intermediary node which knows the identity of the nodes that comprise a given tree. When a new node attempts to join the tree, it generates a local expanding ring search (i.e., one that has a limited TTL value), and also sends a query to the MR. Upon reception of the query, the MR informs nodes on the tree to advertise itself to the node that wishes to join the tree.

While the use of the MR represents promising work in attempting to reduce the cost of the one-to-many join, there are a number of concerns given its current design. One concern involves the amount of potential state that needs to be maintained by this entity. In addition, given the nature of its importance, fault tolerance addressing the single point of failure becomes a critical issue. To address this issue, QoSMIC would need to support an additional server replication protocol if a single primary MR were used for each group. Another approach would be to designate multiple MR's per group.

A more intrinsic concern about MRs involves their designation and advertisement. One of the objectives of QoSMIC is "Limited Impact of Pre-Configuration Decisions" [38]. In the context of previous work such as CBT, PIM, this objective has related to the problem in selecting cores and advertising their existence before a multicast session has come into existence. The a priori selection process has the potential of selecting extremely non-optimal cores, while the

advertisement of <group, core> mappings leads to a potential explosion of state stored in routers. QoSMIC, as well as YAM, address this issue by having the core be a product of group membership. Ironically, however, the use of an MR re-introduces the problem of a priori selection and advertisement. In [38], the authors indicate that all leaf routers in all stub domains know the existence of an MR for a multicast group. To reduce this impact, one can assume that a hierarchy of MRs that advertise a range of multicast group addresses would need to be in place for a such a system to scale. Of course, this adds more complexity in the design and implementation of QoSMIC. Finally, the MR introduces a number of control message exchanges that can contribute to the delay in grafting a new branch onto a tree. This exchange involves:

- *Query message*: Leaf router to MR
- *Advertisement Command message*: MR to tree node(s)
- *Advertisement message*: Node(s) to Leaf Router
- *Join message*: Leaf Router to Node

The last message above is the one that grafts the new branch onto the tree. The topological distance between all the above entities determines the actual delay in grafting a new branch.

Other than the MR, and given that QoSMIC is based on YAM, many of the issues and design goals of QoSMIC and YAM are the same. These shall be discussed in more detail in section 3.2, where I present the design of YAM.

3.2 Overview of YAM

One of the primary functions of the YAM protocol is to provide a means by which multiple paths can be discovered from a joining node to the shared root. Once alternate paths are discovered, a choice can be made to determine which route can best satisfy the desired service of the joining node. The choice can be dependent on the characteristics of the tree (e.g., degree of fan-out of on-tree nodes), characteristics of the network (e.g., number of hops), or by the resource information associated with each path (e.g., minimum delay). In all of these cases, YAM uses protocol primitives to gather information on a hop-by-hop basis from on-tree nodes to the node attempting to graft a branch onto the shared tree. By gathering its own information, YAM removes an dependence on other protocols.

Inherent in it ability to discover multiple paths, YAM operates in a distributed and incremental fashion – as new branches are grafted onto the tree, additional choices are made available to subsequent nodes joining the group. Nodes that attempt to graft a branch onto the tree do not rely on globally distributed information, as in the case of MOSPF. Nor does YAM attempt to find optimal paths in order to graft a new branch. Instead, the design of YAM is based on an on-demand discovery process constrained by a single metric unicast routing protocol.

Another primary feature of YAM involves the on-demand dynamic establishment of a shared core based on group membership. Specifically, the core is elected by having the initial receiver graft a branch onto that node. This removes the need for prior election and system-wide periodic advertisements of cores that are broadcast/multicast throughout the (inter)network. Another benefit in using the Initial-Receiver rule to select the core is that it reduces the potential Steiner tree problem[44] to one of gradually expanding the tree to with newly joined receivers. The Steiner tree problem is essentially a minimal connection problem. In its simplest form, the problem can be stated as: join a set of vertices whose arcs are of minimal weight. The problem, in the same nature of the traveling salesman problem, is NP-complete for graphs comprised of more than just a few nodes. With respect to YAM, the Steiner tree problem is avoided by focusing the tree's construction on the grafting a branch based on a choices of a path, as opposed to all potential neighbor to neighbor links of the network.

- **Terminology**

To facilitate the readers understanding of this section, the following provides a description of some of the terminology used in the YAM specification. In certain cases, terms and definitions are borrowed from previous work.

> Core: This is the node that acts as the root for a shared tree. There is only one active core for a tree, and it determines which nodes are to be the candidate cores in case the current core fails. The selection of the initial core is determined by group membership; i.e., the first receiver that joins a group. This node is also referred to as the shared core and is equivalent to PIM's Rendezvous Point (RP) and CBT's Primary Core.

Candidate Core: This is a node that is selected by the current core so that in the case of failure, a new shared tree can be formed without the need of all nodes to use a one-to-many discovery mechanism. Typically, an ordered list of candidate cores are derived from existing on-tree nodes, with the list being distributed to all on-tree Joining Nodes

Region: This is a topological area that is comprised of a YAM capable routers. Regions can be a subset or superset of (or congruent with) a unicast routing domain. Hence, a region can correspond to a OSPF Area or a BGP Autonomous System (i.e., a unicast routing domain). For the sake of simplicity, one can assume that the boundaries of a region are the same as those of a unicast domain. Finally, Regions are classified as either Transits or Stubs. The former correspond to today's Internet Service Provider (ISP), and the latter correspond to the clients of today's ISPs that do not forward traffic to another domain.

Joining Node: This is a node within a stub region that grafts the internal intra-region to branch to the inter-region shared tree. If the tree does not exist, then by default, this node is elected the shared core for all other receivers. For a given range of multicast addresses, there is only one primary Joining Node used to graft intra-region branches onto the shared tree. However, for the sake of redundancy, there can be several Joining Nodes that are associated with various ranges of multicast addresses. In addition, there can be secondary Joining Nodes that can be used in case the primary fails.

Leaf Router: This is the router or node that receives an IGMP-Join and initiates the construction of an intra-region branch between itself and the Joining Node.

Fan-Out: This term describes an on-tree node in relation to the number of downstream interfaces associated with a group. If the fan-out is described as high, then the node has many downstream interfaces on which it replicates multicast packets and forwards it down the tree.

Any-point Distribution: This term refers to the distribution model of bi-directional tree protocols (e.g., CBT) in that once the tree is constructed, any on-tree node can be used as a starting point for distributing data through the tree. Conversely, single-point

distribution refers to uni-directional tree protocols (e.g., PIM) that only allow distribution to start from the shared root.

3.2.1 Goals

There are several goals that determine the architecture and specification of the YAM protocol and differentiate it from that put forth by CBT or PIM. The five primary design goals of YAM involve the following:

1. Select core based on topology of group membership
2. Constrain configuration information and corresponding periodic advertisements
3. Hierarchy of signaling
4. Discover alternate paths
5. Variety of metrics/criterion to construct a tree

The first item reflects a key design feature of YAM wherein the shared core is initially assigned to the first node that sends a one-to-many Join message -- i.e., a function of the first node that joins the group. The lack of response to the join message automatically elects it as the shared core for the group. The rationale behind this selection is two-fold. First, it offers a measure of simplicity that removes the need and corresponding overhead (state storage) of mapping a shared core with a group address. Second, with a bi-directional tree, the receiver will always act as a sink for multicast data, thus the branch extending from the core will have value as long as a receiver is associated with it. In Chapter 6, I discuss the case when the receiver membership changes and the initially elected core may need to change locations as well.

The 3-way handshake of leaf routers discovering the tree, receiving hop-by-hop unicast responses (with embedded path metrics) from on-tree nodes, and selecting a response, allows the protocol to gather information about the network and the tree in an on-demand basis. However, the consequence of this approach is that the time used to graft a branch is increased from the existing 0.5 Round Trip Time (RTT) to 1.5 RTT. In addition, the one-to-many discovery mechanism introduces an added cost to the network since it generates more control messages than today' s shared tree multicast protocols. A more detailed discussion of how this is accomplished is presented below.

The second goal is related to the first and focuses on the desire to constrain the use and impact of configuration information. In past work [5, 27], the selection of a shared root has been dependent on a priori selection and configuration. In certain cases, this was coupled with periodic advertisements sent to a set of multicast routers so that potential leaf routers would know which shared root should receive join messages. With the use of a one-to-many joining mechanism, YAM has an inherent discovery mechanism used in determining where a root or nodes of a tree exist. Thus, there is no need for system wide configuration of core <=>group mappings.

The third goal, hierarchy of signaling, is an acknowledgment that the use of a one-to-many signaling mechanism can result in a sizable amount of network overhead producing an unwanted cost to the network. As a result, the architecture of YAM can be seen as part of a three-tier signaling structure.

The fourth goal, discover alternate paths, relates to the case in which a default path does not satisfy a type of metric or a target QoS requested by the application. The discovery process is accomplished with a one-to-many joining mechanism, which potentially discovers multiple paths that exist between the joining node and the existing tree. The word potential is used to acknowledge the fact there may exist only one path. In addition, since the ability to discover multiple paths is a function of the number of responses by on-tree nodes, there may exist other paths not discovered by the one-to-many joining mechanism. However, the advantage of my approach is that it can operate over any single metric unicast routing protocol. As the number of group members grow, the number of responses by subsequent on-tree nodes will also increase. Based on a specific metric or criteria, the selection of the response is then used to install state and graft a branch between itself and an existing on-tree node.

This leads one to the fifth goal, which is also related to the first, and focuses on using different metrics or criterion to construct a tree. The forth and fifth goals are defining aspects of my approach in designing a multicast routing protocol that produces QoS sensitive trees. Some of the metrics that I specify in the protocol relate to unicast as well as multicast delivery of traffic (e.g., delay). Others are solely applicable to multicast trees (e.g., Fan Out). The following metrics have been defined in this thesis for the YAM protocol:

1. Bandwidth
2. Delay

3. Hops
4. Fan-Out
5. First-Come-First-Served (FCFS)

The first three criterion are based on information obtained from the perspective of the path that exists from the core towards the joining node (which is assumed to be near the actual receiver). This is a slight departure from existing protocols that install state from the receiver towards the core. This distinction has no effect in the case of the Hop metric. However, Bandwidth and Delay are metrics that can change as a function of load of a specific interface, thereby potentially causing a corresponding change in the path between the two nodes.

The fourth criteria is based on the selection of the path that has a close on-tree node, in terms of number of hops, with a high fan-out. The purpose of this strategy is to try to promote the construction of a tree that exhibits a high fan-out at its leaves instead of near its core. Chapter 5 shows why this strategy can be beneficial in conserving state within the network.

The fifth criteria is a slight departure from the previous four in that it simply chooses the first response that it receives from the one-to-many join. As such, the amount of time needed to construct a branch is minimized since there is no need to compare the first response with any other.

3.3 Architecture

This subsection presents the architectural design of the YAM protocol.

- **Tiered Signaling**

The architectural model of YAM centers on a tiered signaling structure used to graft branches onto a shared tree that can span regions. The reason for using a tiered architecture is to aggregate signaling, such as explicit join messages, at the border of a given region. Currently, all IP multicast protocol architectures follow a 2-tiered model, wherein host-to-network signaling is accomplished with IGMP, and network-to-network is accomplished via a multicast routing protocol. In YAM, this example is extended as three tiers so that joins can be aggregated within a

region. This is accomplished through the local selection of a single Joining Node used to extend an intra-region branch, resident within a stub region, to an existing inter-region branch of a shared tree. Figure 4 below displays an example of the three tiers of signaling.

In this figure, IGMP-Joins for multicast group are localized within an IP subnet and are sent from the receiver to the leaf router. In turn, these routers send corresponding YAM-Joins to a designated Joining Node (JN) for the stub region. Finally, the JN sends a YAM-Join that spans other routing domains; thus, finalizing the construction of a branch on the inter-region shared tree. The key aspect in this high level example is that protocol control messages are aggregated at each boundary {LAN and Stub Region}. By default, the boundaries of the stub region will be the same as a stub unicast routing domain. This allows the initial deployment of YAM to follow the existing inter- intra-domain routing split. However, through configuration, a stub region could include several stub domains, or it can be as small as a single subnetwork within a routing domain.

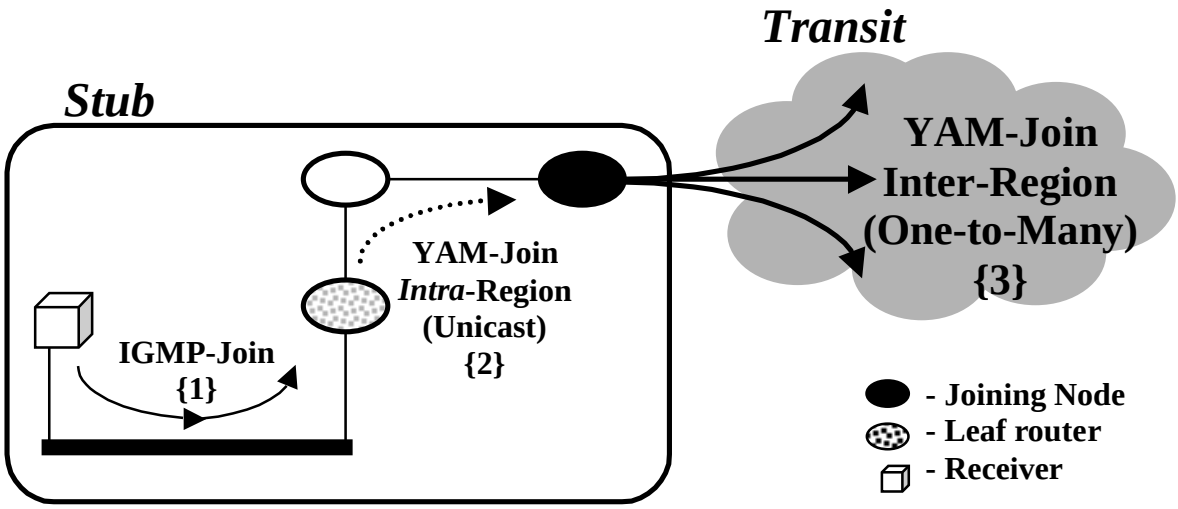


Figure 4 - Example of a Three Tier Join

Another reason for a multi-tiered model for YAM is to map the protocol along the same boundaries of unicast routing protocols used in today's Internet. This allows the protocol to take advantage of local authority over resources within a stub region so that a priori configuration information is defined and propagated locally -- thus, constraining its impact on a system wide basis. In the case of Figure 4, the selection of a designated Joining Node can be accomplished in one of the following ways:

1. Configuration Files
2. DNS Updates
3. YAM Advertisements

The first option involves static configuration files that are loaded in each YAM-capable router within the stub region. This approach has the appealing aspect of no additional overhead generated by the YAM protocol. However, it has no scaling capabilities or means of insuring that information is consistent.

The second approach is an out-of-band mechanism that takes advantage of an established resolution mechanism that can be dynamically updated [94]. Initially, the database of the DNS server is configured with mappings of multicast groups with a list of Joining Nodes associated for the stub region. When a YAM multicast router is booted up, it makes a query to a DNS server to obtain the existing JN<=>group mappings. Over time, the listing may be updated by DNS update messages. It can be expected that an additional DNS record will need to be defined, but no additional DNS protocol primitives will need to be specified. The biggest appeal of this approach is that information can be provided on an as needed basis via the query-response architecture of DNS.

The third approach involves the use of in-band periodic YAM messages that advertise the presence and identity of a Joining Node. This type of advertisement can be considered a region-wide report.

- **Building a Branch: The 'Pull' Model**

In tandem with the three-tier signaling structure, the YAM architecture relies on a "pull" model that involves an on-demand request of information from the tree in order to graft an inter-region branch onto it. This "pull" model is initiated by the Joining Node sending a query in one of three ways:

1. Unicast: point-to-point message sent to a specific unicast destination,
2. One-to-Many: a message sent via reverse path broadcast to all inter-region YAM routers

The first type of query is used when the Joining Node has some knowledge about the location of on-tree nodes or the candidate root of a tree. This type of joining mechanism can be considered the same as that used by PIM and CBT; all multicast routers have knowledge about the identity of the tree and its shared core.

The second type of query focuses on discovery and is used when there is no a priori knowledge of the location of an existing tree or its shared core. The discovery process is accomplished by distributing a one-to-many join message from the Joining Node via reverse path forwarding. When an on-tree router receives the spanning join, it terminates the distribution process and sends a unicast response back to the Joining Node. By default, these responses are sent hop-by-hop, with each node on the path being recorded. This recording is used to gather information on specific metrics and helps ensure that routing loops are not formed. Based on the responses received, the JN sends a unicast join that selects a given path and completes the three-way handshake used to graft a branch onto the shared tree. If the JN does not receive a response from the one-to-many join message, then by default, it becomes the shared core for the next receiver to join the group. The following figures presents a simple depiction of grafting a branch using the one-to-many join.

In Figure 5.a, there are four stub regions that are already connected to the target multicast tree. A Joining Node in region {A} sends a one-to-many join message that is distributed through the transit region and received by nodes {1 & 2} that are on the target multicast tree.

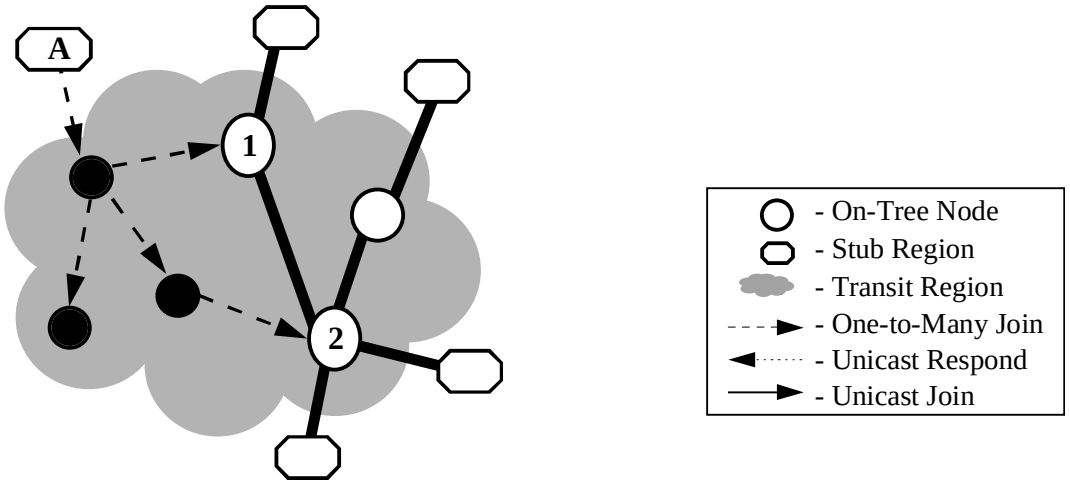


Figure 5.a - One-to-Many Join

Figure 5.b shows that nodes {1 & 2} send a unicast response back to the Joining Node within region {A}. This response is sent hop-by-hop and is updated with the address of each node traversed along the path. After receiving two Respond messages, figure 5.c shows that the Joining Node in Region A issues a unicast join message towards node {1}. This is the third part of the YAM three-way handshake, which is used to select one of N number of paths. With this confirmation, data is allowed to flow downstream through the new branch, and state along the other potential paths is timed out. Periodic unicast join messages are sent upstream on a hop-by-hop basis and are merged by each parent in order to limit the amount of system overhead experienced by the existing tree.

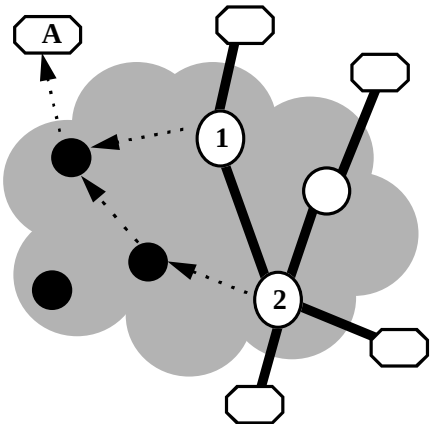


Figure 5.b - Unicast Response

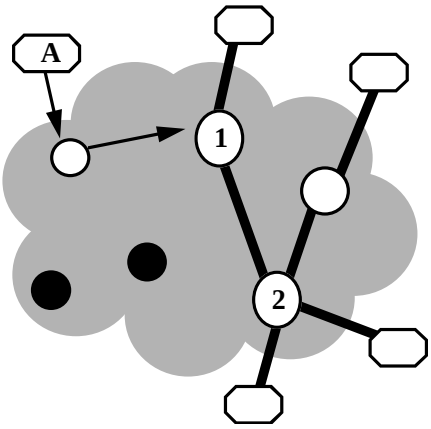


Figure 5.c - Unicast Join

Conversely, periodic responses are issued by the shared core and distributed to each downstream on-tree router in order to refresh existing pinned softstate. When the Joining Node discovers a break in upstream connectivity, or when state times out (no response received), it issues another join message. If previous responses were cached, then a unicast join is sent to a specific destination. Otherwise, a one-to-many join is sent to discover the current location of the tree.

- **Loop Suppression**

As described in [24], one of the appealing characteristics of using RPF is that (in the normal case) it does not require persistent state, such as sequence numbers, to prevent loops. This is because received traffic is only forwarded to the other interfaces when the unicast routing tables indicate

that the inbound interface is the shortest path to the source of the traffic. However, I refer to this as the 'normal' case, because there can exist conditions where unicast routing tables of node(s) are temporarily inconsistent and form temporary loops, which can cause localized multicast storms. These storms are eliminated either by a convergence in routing tables and/or through the expiration of the control packet's TTL.

In its favor, Joining Nodes only send a single one-to-many query-join message, as opposed to a stream of multicast data packets. In addition, the convergence time of most unicast routing protocols tends to be in the order of seconds. While this reduces the impact and severity of the multicast storm, it does not eliminate it. As an added measure to deal with this problem, YAM nodes have the option of temporarily caching information that uniquely identifies a one-to-many query-join. This information is comprised of the source address, target multicast address, and timestamp; the latter two items being stored in the query-join control message (see section 3.2.4 for more details).

It is stressed that this additional information is stored temporarily (e.g., 1 minute), and then removed from the cache. In this manner, YAM does not retain **<source, group, timestamp>** information for the entire system, but rather a finite set of entries attributed to control messages for a short period of time as added protection against multicast storms.

- **Asymmetric Paths**

The ideal distribution scheme involves shared trees whose branches are comprised of symmetrical paths between a leaf node and the shared core. However, due to policies (or certain types of subnet technologies like satellite links), the path between a Joining Node and the shared core may be asymmetrical. If it is, then all nodes along the candidate branch must conform to single-point distribution scheme, wherein all source traffic must be encapsulated and forwarded to the shared core for distribution through the shared tree. Conversely, if the path is symmetrical then any-point distribution can be used (i.e., distribution of source traffic can begin at any on-tree router).

Figure 6 presents an example in which one branch has a set of asymmetric unicast paths between node {1} and the shared core. Specifically, traffic received from the core by {1} travels along a different path than traffic sent by {1} towards the core. Therefore, in order to prevent loops, it is

necessary for node {A} to encapsulate multicast data originated from {1} and forward it to the core. The core then decapsulates the traffic and distributes the multicast data down the tree. In this example, the core acts as the single point of distribution for the tree.

The unshaded areas of Figure 6 represents unicast paths between nodes; as shown by the entire right branch emanating from the core. Thus, multicast traffic emanating from node {2} is immediately distributed by node {B} to the other on-tree nodes without the need to encapsulate the traffic and send it to the core.

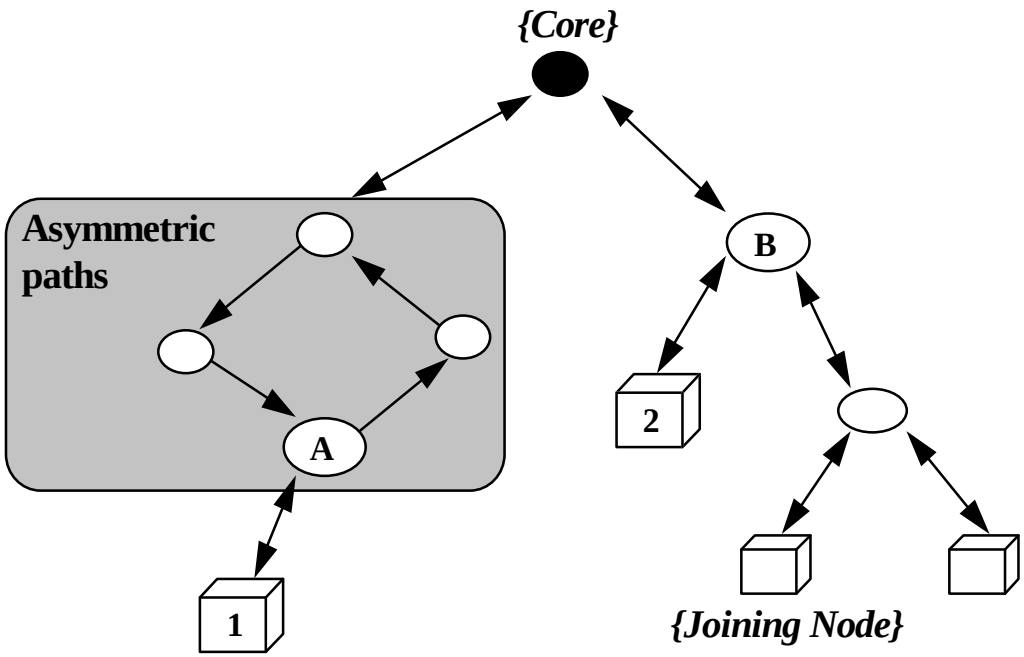


Figure 6: Tree that has Asymmetric & Symmetric branches

An earlier version of YAM [12] suggested that in cases where the network supports RSVP, a Path message containing an AdSpec object can be sent following the transmission of a YAM Respond message. This RSVP message could inform the Joining Node about the available resources of that path. However, it may be possible for a downstream node to forward the RSVP-Path message before the YAM-Respond message. This transposition could lead to a race condition in which the Joining Node is not informed about the available QoS between itself and the on-tree node, thus concealing information in choosing the best path for the new branch. For this reason, and the potential reliance on RSVP to provide QoS information, I have chosen to have the YAM-

Respond be solely responsible for informing the Joining Node about existing QoS information; from itself to the on-tree node, and possible from another on-tree node (e.g., the core) to the on-tree node initiating the Respond message.

- **Unsolicited Information: The 'Push' Features**

The YAM protocol incorporates the capability of distributing unsolicited information to on-tree or off-tree nodes. This feature represents an integration of a "push" model with the baseline "pull" architecture of YAM. In the general case, the push capability incorporates an additional level of versatility in having nodes of a tree distribute unsolicited information about itself. The target of the pushed information can involve nodes within a region (i.e., domain-wide reports), nodes that reside in neighbor domains, and finally on-tree nodes of a given tree. The realization of the push model involves the distribution of Information messages. The form of the distribution is dependent on the type of message being sent.

Currently, there are two types of Information messages. One involves the optional periodic advertisement of a Joining Node within a domain. This can be considered a domain wide report that informs leaf routers of the identity of the target Joining Node with which the initial intra-region branch is built. The distribution model is based on Reverse Path Forwarding via a reserved well-known multicast address.

Another type of Information message involves the identity of an ordered list of secondary cores -- nodes that can act as the primary core in case the current core fails. This information is only distributed by the primary core to on-tree nodes of the inter-region shared tree. By advertising secondary cores, the tree makes itself more robust in an effort to avoid a single point of failure. The distribution model for this message involves hop-by-hop unicast to all downstream routers on the tree.

- **Non-Member Senders**

Previous sections have focused on grafting a branch from a receiver to an existing tree. This process involves an a priori exchange of control messages before any data is sent to the receiver. In cases were the receiver is also a source of traffic (i.e., a member sender), the process also informs the receiver of the address of the on-tree node used to distribute its source data down the

tree. Hence, the issue of how to distribute traffic from member senders is inherently solved by the explicit joining mechanism used by YAM.

However, the case of non-member senders is different because the need to forward traffic from an off-tree node onto the tree is data driven. Thus, there exists a strong motivation to expedite the discovery of the tree and its distribution point(s), so that source data does not have to be stored in buffers for significant periods of time. Within the context of the YAM protocol, the problem space, and subsequent solution, is divided along the two tiers of signaling. Note that the following assumes that no cached information exists within the leaf router or in the Joining Node.

In the case of the first tier (i.e., building an intra-region branch within the stub region that contains the receiver), the leaf router already knows the identity of the Joining Node used to forward source data and graft branches on the tree. Hence, the source data packet is encapsulated within the unicast query-join message and forwarded to the region's Joining Node. All subsequent data packets are sent natively along the path established by the initial encapsulated source packet. In addition, periodic join messages are sent to refresh the existing state. When the non-member sender stops sending source data, the join messages are halted and the path is eventually timed out.

In the case of the second tier, the actions of the Joining Node are determined by whether or not it has a cached mapping of the target multicast address and the shared core. If the mapping exists, then the initial multicast source data is encapsulated in the same manner as before and forwarded to the unicast address of the shared core. Subsequent data packets are then sent natively. If the mapping does not exist, then the Joining Node must issue a one-to-many join to discover the location of the tree. In this case, the response from the one-to-many join involves control message(s) that contain the identity of an on-tree node used to distribute traffic through the tree. The consequence of this latter action is that the initial source data must be buffered for (1) RTT by the Joining Node.

3.4 Protocol Overview

3.4.1 Query Type Messages

Query messages are used to discover information and issues requests. The most common request involves issuing a join that grafts a branch onto a shared tree.

- **Unicast Join**

This message is unicast to a specific destination. Typically, this type of join message is used within a stub region to establish an intra-region branch between a leaf router and the designated joining node. If the leaf node is a non-member sender, then the Query-Join message installs state towards the destination; thus, installing a unidirectional path from the source to a node that distributes traffic onto the multicast tree. Respond messages are sent back to the leaf router indicating that the designated joining node is active.

If the leaf node is a receiver (meaning that it has sent an IGMP-join), then the YAM Query-Join, and corresponding Respond message, installs state in both directions so that the leaf node can send and receive multicast data along the same path. By default, this intra-region branch is a symmetrical path.

It should be noted that the Query-Join message is sent hop-by-hop and is subject to whatever underlying unicast metric is used to forward traffic from source to destination -- this typically being the shortest hop metric. However, if the intra-region routing protocol is capable of providing multiple paths, then YAM will take advantage of this capability.

Finally, if the multicast group is locally scoped to that region, then the destination (Designated Joining Node) can be considered the local core for that group. Thus, no attempt is made to establish an inter-region branch. However, if the group has a global or inter-region scope, then the destination node makes an attempt to graft the intra-region branch onto a tree that spans other regions.

- **One-to-Many Join**

The one-to-many Query-Join is part of a 3-way process that involves sending a discovery message, receiving response(s) from nodes on the tree, and selecting a response. The end result being that a potential choice of paths can be made available to the joining node in terms of grafting a branch onto the tree.

When the location of the shared tree is unknown, a joining node sends a one-to-many Query-Join message using Reverse Path Forwarding to forward the message through the network. This type of distribution is used as an on-demand means of discovering the existence of a tree for a given multicast group. In a previous multicast protocol [26], the inherent discovery mechanism of RPF is used to install state in the network and build a data driven delivery tree from the source to all the receivers of a group. However, YAM only uses RPF as the first part of the 3-way handshake to discover an existing tree and does not use this control message to store state in the (inter- or intra-region) routers of a network.

When a node on the tree receives the one-to-many join, it discontinues the RPF procedure and sends a Respond message to the joining node. This initial response performs three tasks: a) it installs temporary softstate downstream from the tree to the joining node, and b) it gathers a measure of available resources on that path (e.g., an approximation of available bandwidth and accumulated delay from the core to the joining node), and c) the route is recorded in the event that asymmetrical paths exist between the on-tree node and the joining node.

By default, the Respond message is sent as a hop-by-hop unicast message with no explicit source routes being used. This approach assumes that if the group membership of the tree is topologically disperse, then several default paths should be discovered by the joining node. However, since these paths may not be disjoint, it is necessary to record the exact downstream route.

3.4.2 Respond Messages

Respond messages are only sent in response to query messages and are typically used to install temporary state along a given path. This temporary state is refreshed through a periodic exchange of query/response messages. When this exchange terminates, or follows a different path, old state is timed out. It should be noted, though, that the selection of a different path is determined by the joining node (via the query/response mechanism) and not through the existence of a "better" or more optimal path as determined by the underlying unicast routing protocol.

The Respond message can also be used to inform the destination as to whether or not the path from the joining node to the shared core is symmetrical. If the path is symmetrical, then the branch can be used for any-point distribution; wherein, any on-tree node of the branch can be

used as the initial point of distribution for source traffic. However, if there is a segment of the path that is asymmetrical, then the branch is treated as single-point distribution. In this latter case, all source traffic must be encapsulated and sent to the shared root, which is where the on-tree distribution of multicast traffic begins.

3.4.3 Information Messages

Information messages are used to convey unsolicited information to other on-tree or off-tree nodes. Currently, there are four information control messages. One type of message involves the advertisement of <core, group> mappings by Joining Nodes. This message is distributed via reverse path broadcasting and is scoped with a TTL value so that only topologically close neighbors receive it. The purpose of this message is to update the <core,group> caches of other off-tree Joining Nodes in order to help minimize the delay in forwarding traffic from non-member senders.

A second type of Information message involves the periodic advertisement of candidate shared cores. This message is sent by the existing core to all the on-tree joining nodes and contains an ordered list of on-tree routers that are designated as potential shared cores for the tree. If the existing core fails, then the Joining Nodes shall attempt to graft a branch onto the first router of the ordered list. If this candidate router has also failed, then the next router on the list shall be used. This type of information message is sent periodically and on an infrequent basis.

The third type of Information message is distributed on a periodic basis via RPF and is targeted to all the YAM routers within a stub region. The message contains mappings of multicast addresses to specific Joining Nodes so that leaf routers know the identity of the JN on which to graft an initial intra-region branch. The grafting is then accomplished by unicast joins.

The fourth type of Information message is distributed locally within a LAN and is used to elect a designated router for a broadcast multi-access subnetwork. This message is periodically broadcast with a TTL of 1 and contains a preference used as the basis for selecting a designated router. If the preference field is null, then the selection is based on the lowest IP address of YAM routers attached to the LAN. Otherwise, the field can be used to reflect characteristics of existing reservations -- such as minimal queuing delay.

3.4.4 Connectivity Failure

In the event that a link or upstream node has failed, the downstream node can be informed in one of two ways. The first involves monitoring of unicast change-of-link-status. If this interaction between the YAM protocol and the underlying unicast routing protocol is not possible (or implemented), then the YAM node can determine a change of link status via a failure to receive a periodic Information message sent by each parent node in order to refresh downstream state. In either case, the downstream YAM node issues a one-to-many join in order to discover other paths to the shared tree. Note that the recorded route feature of each Response message ensures that the Joining Node discovers if any Response message contains a loop.

3.5 Protocol Specification

This section provides a detailed description of YAM control messages. All YAM control messages are divided into 2 parts: the Fixed Header segment and the Primitives segment. The Fixed Header contains generic information about the YAM control message and is comprised of fixed sized fields that are never expected to change during the evolution of the protocol. The Primitives segment of the control message is one of 3 different types: Query, Respond, and Information. Each has a distinct format of fixed fields and an optional number of variable length fields -- the latter being in the form of Type-Length-Value (TLV).

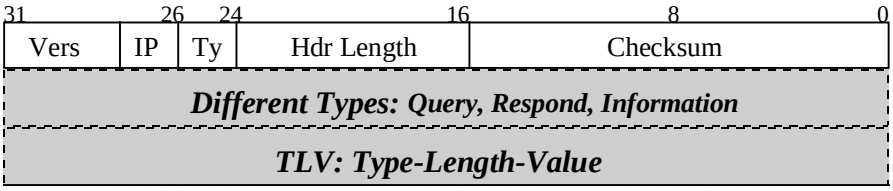
There are two items that should be noted when reviewing the following specifications of a YAM control message. One is that all fields titled Reserved are reserved for future use and must be ignored by receivers and set to zero by the sender. The other is that for TLVs, a Type of {0} indicates that no value is used and the Length field must also equal {0}.

Below, I present the Fixed header, as well as its ordered relationship with the Type segment and the TLV segment of a YAM control message. I also present the format for each of the different segments (i.e., Query, Response, Information). In addition, each of these sections identify the different TLVs used for a given segment

It should be noted that nearly all control messages are sent using unicast destination addresses in the IP header. The only exception involves the *one-to-many* Query message, which is designed

as a dynamic discovery mechanism for YAM and uses a reserved multicast address of 224.0.0.20 (tentative).

- **YAM Fixed Header**



Vers: (4 bits) - Version of the YAM protocol. Initial Value set to 1

IP: (2 bits) - Identify the version of IP.

Value: 0 - IPv4, 1 - IPv6

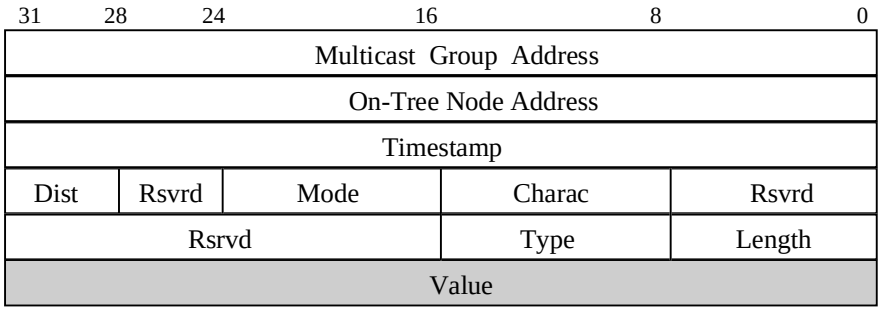
Type: (2 bits) - Field identifies the type of YAM control message

Value: 0 – Query, 1 – Response, 2 – Information

Hdr Length: (8 bits) - length of the entire YAM control message. The size includes the Fixed Header segment and the Type Segment and is specified in terms of the number of octets.

Checksum: (16 bits) - checksum of the entire YAM control message.

- **Query**



Group: (32 bits) - Target multicast group address that is to be joined.

On-Tree Node: (32 bits) - Unicast address of an on-tree node. This value is set when the message is to be subcasted to a constrained topological region.

Timestamp: (16 bits) - Timestamp of when the packet was created. This value is used as a tie breaker when two or more Joining Nodes attempt to establish the initial inter-region branch (and shared core) of a tree at roughly the same time. The lowest timestamp value determines which Joining Node is to be the initial shared core for the inter-region group. It should be stressed that the value of the timestamp can be grossly incorrect.

Distribution: (4 bits) - This field specifies the manner in which the join message is to be distributed.

Value

0. Unicast: A specific unicast destination address is used in the IP header. No state information is setup along the path between the source and the destination.
2. One-to-Many: The reserved multicast address, 224.0.0.20 (tentative), is used as the destination IP address. The message is distributed via Reverse Path Forwarding from the source.
3. Constrained One-to-Many: The reserved multicast address, 224.0.0.21 (tentative), is used as the destination address. Unlike {2 - One-to-Many Join}, the distribution of this message is constrained by having a specific unicast destination target that is coupled with a routing metric.

Mode: (8 bits) - This field defines the state of the query message. The three values are:

Value

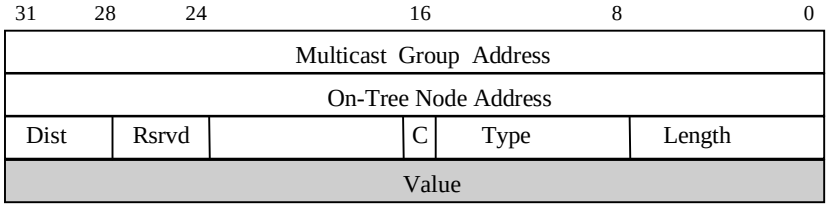
1. Join: This is a unicast message that is sent to graft a branch onto a shared tree.
2. Request Candidate Cores: This message is sent by the existing shared core in order to discover the identity of other on-tree routers that are not simply immediate neighbors. Based on the response, the shared core will then select an ordered set of candidate shared cores in case the existing core fails.

Characteristics: (4 bits) - This field defines the characteristics of the requested branch.

Value

1. Unidirectional: The request is for a unidirectional branch from a non-member sender to a distribution point on the tree.
2. Symmetric: The request is for symmetric paths between the node and the tree so that the branch can be used for any-point distribution
3. Any: The request is for any type of branch (asymmetric / symmetric)

- **Respond**



Group: (32 bits) - Target multicast group address that is to be joined.

On-Tree Node: (32 bits) - Unicast address of an on-tree node. This value identifies the shared core of the multicast group.

Distrib: (4 bits) - This field indicates the manner in which the response is to be sent to the node that issued the Join message. The three values applicable to the message are:

Value

1. Unicast Hop-by-Hop: The response is sent as a unicast message and hop-by-hop towards the originator of the Join message. Each node along the path installs state, updates the message, and forwards it to the next hop. This type of distribution is a typical response for query-joins sent on behalf of a receiver. This value indicates that the **{S, P}** bits must be followed.

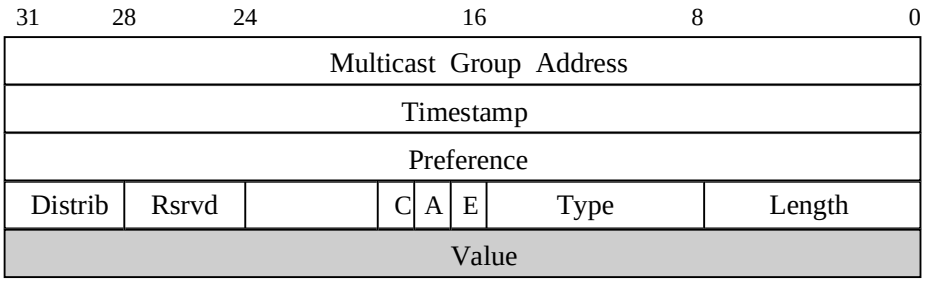
Flags: (8 bits) - This field contains several flags that specify the characteristics of the respond control message. The values of certain flags determines which TLV must be used with this message.

{P} Path Bit:

{0} - a value of (0) indicates that the path between the originator of the Join message and the shared core is symmetric.

{1} - A value of (1) indicates that the path is asymmetric. Once the bit is set to one, it cannot be altered/reset.

- **Information**



Group: (32 bits) - This field contains the multicast group address of which this message pertains to.

TimeStamp: (32 bits) - This field contains a timestamp of when the message was created.

Preference: (32 bits) - This field indicates the precedence value of the message. Currently, this value is used to elect a designated router for a broadcast LAN.

Distribution: (4 bits) - This field specifies the manner in which the information is to be distributed from the perspective of either point-to-point or one-to-many. The field also indicates whether the information is targeted to an node or whether the information is only to be distributed to on-tree nodes.

Value

0. Point-to-Point: The message is to be sent as a unicast message

1. One-to-Many: The message is sent in a one-to-many basis using Reverse Path Forwarding from the source. Typically, the messaged will be scoped with a TTL value set in the IP header. The default TTL value is 5.
2. On-Tree Nodes: The message is to be sent to on-tree nodes of a given multicast address. This message must have a TLV that identifies the target destination multicast address. This message is typically scoped with a default TTL value of 5 set in the IP header.

Flags: (8 bits) - This field contains flags that indicate the type of information that is being propagated by the Information control packet. Only one bit can be set per message. Hence, all values of zero indicate that the flag is to be ignored.

{C} Candidate Cores Bit:

{0} - Ignore the Candidate Core flag

{1} - This message contains an ordered list of candidate cores. If the current core fails, existing Joining Nodes are expected to attempt to graft a branch based on the ordered list contained in this message. Thus, if the first node of the list is also unreachable, then the next node of the list is attempted.

{A} Advertise Joining Node Bit:

{0} - Ignore the Advertise flag

{1} - This message is used to advertise the designated Joining Node of a stub region. The message contains the IP of the Joining Node, a precedence field, and a range of multicast addresses that are associated with the joining node.

{E} Elect DR Bit:

{0} - Ignore the Election flag

{1} - This message is used to elect a designated router in a broadcast multiple access local area network. The message contains a Precedence field and always has a TTL of 1.

- **Specification of TLV**

The following TLVs have been defined for the YAM protocol. Initially, the majority of TLVs are targeted towards IPv4 networks. Additional TLVs can be defined for IPv6. In the following sections, Octets 1 through 4 correspond to an IPv4 address in the form of A.B.C.D. In cases where the length field has a specific value associated with it, a fixed number of addresses can be associated with the TLV. In cases where the Length field is associated with a value of "N", then an arbitrary size is associated with the TLV.

Record Route

		Type = 4	Length = N
1'st Octet	2'nd Octet	3'rd Octet	4'th Octet

The result of dividing the length value by 4 indicates how many unicast addresses have been recorded. Note that new addresses are added at the bottom of the list.

Source Route

		Type = 6	Length = N
1'st Octet	2'nd Octet	3'rd Octet	4'th Octet

The result of dividing the length value by 4 indicates how many unicast addresses are in the source route. Note that the list is a strict source route of YAM capable routers.

List of Candidate Cores

		Type = 8	Length = N
1'st Octet	2'nd Octet	3'rd Octet	4'th Octet

The list of addresses is placed in a prioritized manner denoting the order in which other nodes should be considered the shared core in the event that the existing core fails.

Bandwidth (From Core)

		Type = 10	Length = 4
1'st Octet	2'nd Octet	3'rd Octet	4'th Octet

The highest available bandwidth from the unicast path between the shared core to the Joining Node. A variation of this *type* can be realized as available bandwidth along the unicast path from the on-tree node initiating the Respond message to the Joining Node.

Delay (From Core)

		Type = 12	Length = 4
1'st Octet	2'nd Octet	3'rd Octet	4'th Octet

Cumulative delay of the from the unicast path between the shared core to the Joining Node. A variation of this type can be realized as cumulative delay along the unicast path from the on-tree node initiating the Respond message to the Joining Node.

3.6 YAM Metrics, Applications, and RSVP

Given YAM's design, one is provided with a mechanism that can attempt to satisfy the QoS of an application – assuming there exists a path with available resources. However, given such a mechanism, the issue remains regarding its overall integration with QoS capable applications and the existing resource reservation model defined by the IETF for IP based communications.

Figure 7 below shows an example of a set of control message exchanges used to graft a QoS sensitive branch onto a shared tree. The perspectives shown represent the Application, its abstract communication with a YAM Joining Node (JN), and the JN's communication with the network. The sequence of events are as follows:

1. The application sends an IGMP-Join message to the network. Depending on the configuration of the network and boundaries the region containing the application, this "join" signal is eventually propagated to the JN.

2. Assuming the JN has no state for the target group, it issues a 1-to-Many YAM-Join to the transit regions of the network.
3. The Network then sends response(s) depending on the number of existing on-tree nodes.
4. Assuming RSVP is available in the network, RSVP-Path messages are sent towards the JN. These messages contain the characteristics of the type and amount of resources to be reserved. The JN then selects the path that best satisfies the QoS parameters stipulated in the Path message, and grafts the branch onto the tree.
5. The Path message is forwarded to the host containing the application, which in turn issues an RSVP-Resv message towards the JN. The JN forwards the Resv message up the new branch, which contains the signaling information used to reserve resources.

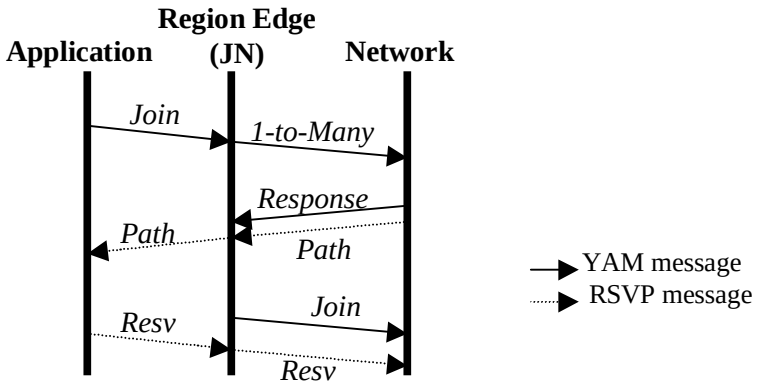


Figure 7: Example of YAM & RSVP control message exchange

Given that RSVP has the ability to convey changes in QoS (e.g., more/less bandwidth), YAM has a corresponding ability to attempt to alter its tree structure and search for a different path upon which to graft a different branch. In the case where less resource is needed, YAM makes no changes to the existing tree. But in the case where more resources are needed (and the existing branch cannot satisfy the request), then the JN can be configured to search for new paths to see if a different one has more resources.

Finally, in the event that RSVP is not available, YAM selects the path that corresponds to criteria or type of metric that has been configured for the JN.

3.7 Implementation

Given the specification above, I present information about my implementation and several issues that arose in implementing the YAM protocol. Many of these issues and the approaches I took in solving them can be applied to other shared tree protocols like CBT, and/or QoS multicast protocols like QoSMIC. My development environment consisted of:

- Gated Daemon (GateD) - Version 3.6 {Merit}
- Multicast Route Daemon (MrouteD) - Version 3.5.1.2 {Xerox-PARC}
- Sun Microsystems Operating System (SunOS) - Version 4.1.3 {Sun Microsystems}

GateD is a portable routing development platform. It is designed to operate over a variety of host operating systems and is used to add IP routing functionality to a host. GateD supports a variety of unicast and multicast protocols; such as BGP, OSPF, DVMRP, and PIM. I developed a version of YAM based on the specification presented above and added it to GateD.

Concerning support for multicast, GateD uses the kernel modifications produced by Bill Fenner of Xerox-PARC in support of MrouteD. These modifications focus on accessing the Multicast Routing Information Base (M-RIB) as well as the Virtual Interfaces (VIFs). The M-RIB is used for determining the extent in which data is replicated and sent to downstream children of the multicast tree. The VIFs contain interface specific information and where designed to be generic enough to support tunneling of multicast traffic through routers that do not support multicast routing. The MrouteD kernel modifications have been ported to a number of operating systems; such as SunOS.

- **Issues**

Given that MrouteD is designed to construct source based trees, the accompanying kernel modifications have been designed to store information for both the source and the group IP addresses. PIM implementations also use the MrouteD kernel, but instead of storing the source

IP address, they store the address of the Core[14], thus no changes are needed to the kernel to support PIM implementations. This address is then used to perform RPF checks to help ensure that a loop has not occurred and that data is only distributed down the tree from the core.

Since YAM is a protocol that supports any-point distribution and builds bi-directional trees, there is no need to perform RPF checks based on the location of the core and its relationship in how the data is distributed through the tree. Thus, a number of code segments in the kernel (specifically, ip_mroute.c) that manipulate the Multicast Forwarding Cache (MFC) structures need to be commented out. In my augmented version of GateD, I populate the MFC structures with group IP address and store a wildcard address of (0xe0eeeeee)[15]. This allowed me to retain most of the existing kernel routines without having to do a major code revision. Any subsequent 'gets' for a group are accompanied by the wildcard address.

Another implementation issue stems from a fundamental design paradigm of using Designated Routers to accomplish IP multicast over broadcast LANs. A fundamental characteristic of Designated Routers is that they are chosen independent as to the QoS they can provide to downstream nodes. A detailed discussion of this problem space is presented in chapter 4.

Note: The implementation of YAM was accomplished to obtain 'real-life' information regarding design issues and applicability of YAM's protocol specifications. In other words, I attempted to gain information regarding the design of a protocol and its theoretical foundation versus the engineering and application of a protocols and its underlying algorithms. From this information, I refined the YAM protocol and applied this knowledge in its simulation (presented in Chapter 5). Since my testbed environment only consisted of five workstations running Gated/YAM, I needed to rely on simulation in order to examine YAM's QoS capability, such as its scaling property, in an in-depth manner.

3.8 Summary

In this chapter, I have presented a QoS sensitive multicast routing protocol, titled YAM, that uses a one-to-many joining mechanism to construct a shared tree. This mechanism acts as an on-demand discovery of where the tree is in the network, the identity of its shared core, and the

[14] PIM uses the terminology Rendezvous Point (RP), but we shall refer to this node as a core for the sake of consistency.
[15] This is an arbitrary value chosen for test purposes only.

potential existence of multiple paths from a new branch can be grafted onto the shared tree. The selection of a path is based on which one provides the best match for the target metric (e.g., lowest delay from leaf to core, highest bandwidth from leaf to core, closest on-tree node that has the highest Fan-Out).

In having a choice of paths in which to construct a tree, leaf nodes are able to tailor the trees construction based on a variety of metrics, as opposed to today's de facto metric of choosing the shortest path from the leaf to the shared core. In addition, the Query/Response architecture of the protocols allows it to continually learn about the dynamics of the tree and alter its structure in accordance to different criteria, such as current topology of group membership. Two of the biggest strengths of the YAM protocol are that it: a) selects its shared core based on group membership, and b) it operates over any unicast routing protocol that calculates a single path to a destination. Hence, YAM does not need an underlying multi-metric routing protocol to discover alternate paths between two edges on a graph. Rather, it relies on different paths being generated by different on-tree nodes. The metric, and its value, is stored and updated by each hop traversed by the response to the one-to-many join. In order to ensure that loops are not formed in the discovery/selection process, these responses are embedded with a recorded route of each hop that comprises a path.

Chapter 4

Issues Related to QoS Multicast

Introduction

This chapter presents a number of issues related to Quality of Service multicasting in terms of its design and implementation within the existing IP multicast framework and service model. I discuss an example of how QoS multicast can be beneficial while at the same time show how the ability to select one of several paths can prove problematic in broadcast LANs that use a Designated Router to forward all data and control messages. To address this problem, I introduce the concept of a Target Router that can be used to identify one of several possible parents in a broadcast LAN.

This chapter also presents some specific issues related to YAM and QoSMIC. One issue involves designs used to decrease the impact of the on-demand discovery process inherent in both protocols. Specifically, I show several different strategies in which the goals and impact of a one-to-many join can be constrained, and present pros and cons of each approach. Another issue involves the support of multiple sources sending traffic at the same time. This particular case represents an arbitrary condition in which the number of sources constitute a form of "some-to-many", as opposed to my previous classification of the many-to-many model in which all group members act as a source of traffic.

Finally, I discuss the issue of core migration. This is a topic that takes into account the dynamics of group migration and its potential for fostering a need to migrate the shared core to a more optimal location. In this discussion, I introduce an optional augmentation to the baseline design of YAM such that under certain conditions, it can migrate the core with no packet loss or duplication. Simulation results of this augmentation are presented in Chapter 6.

4.1 Subnet Connectivity and Broadcast LANs

On-tree nodes can be members of Non-Broadcast Multiple Access (NBMA) subnets, point-to-point subnetworks, or members of a shared broadcast LAN. Given the physical point-to-point nature of the link connectivity, the first two cases are generally not exposed to the situation in which multiple parents can exist from the same input interface. I qualify this statement by acknowledging that certain NBMA technologies can be altered to emanate broadcast transmissions [3, 91]. But in having done so, the subnetwork has reclassified itself and its routers must support responsibilities attributed to broadcast networks.

One such responsibility is the selection of a designated router (DR) [64]. Within the context of multicast routing, the DR ensures that there will never be multiple on-tree nodes sending downstream data into same the broadcast network – resulting in duplicated data sent to the same children. The criteria used in the election of a designated router, as well as its exact responsibilities, is dependent on the multicast protocol. For dense mode algorithms, the DR acts as a designated forwarder in cases were 2 or more nodes have downstream interfaces that are connected to the same broadcast LAN. In this case, the sole focus of the DR is to prevent the generation of duplicate packets. For sparse mode algorithms, the DR has the added functionality of being the node that forwards explicit joins (from protocols like CBT and PIM) to the shared root.

In Figure 8 below, I present an example of a shared broadcast LAN environment that has two connections to the Multicast Backbone (Mbone). The figure depicts a host sending an IGMP-join down its subnet interface, which in turn is received by both routers on the LAN. The figure also shows initial multicast data (before any prunes) being sent to both routers of the LAN – both of which are of equal distance to the source. In this case, router {B} has the highest IP address and is elected as the DR (or more precisely, the designated forwarder) for the LAN, and thus forwards the data on its downstream interface (10.1.1.2).

A key aspect related to the figure is that no assumptions are made by the host as to which router is to receive and process the join. Nor does the IGMP-join process influence which router is to be elected as the DR. This can be considered beneficial from the perspective of isolating information about the network from the host. But in other contexts (as presented below in subsection 4.1.2), the selection of a single DR can also act as a limiting feature.

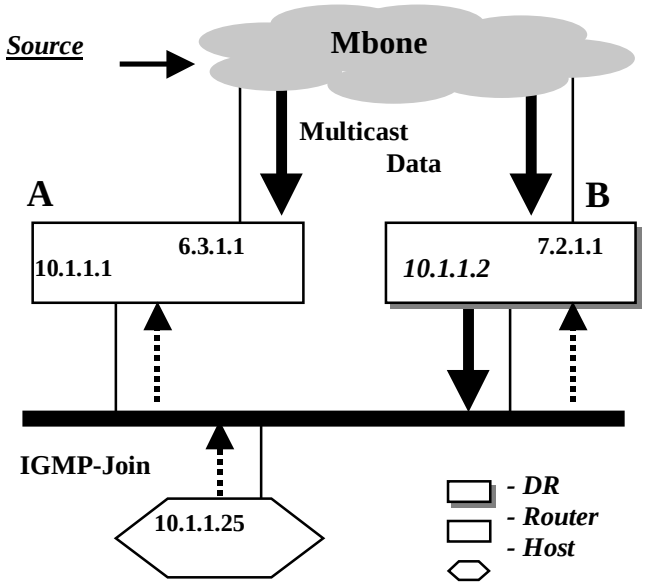


Figure 8: On-Tree Nodes on a Broadcast LAN

- **Forwarding Model**

The forwarding model of IP multicast is predicated on the basis of the input interface that receives the multicast data. If the input interface has state for the group, and the interface is (or can be) marked as upstream, the data is then forwarded down all the downstream interfaces. The key feature of this model is that the acceptance of multicast data is based on the input interface, as opposed to a combination of the IP address of the previous hop and the input interface. This is because the DR enforces the rule of a single parent for each child of the tree on a broadcast LAN. An appealing aspect of this model is that the process of determining a parent is based on a single atomic element: the input interface.

4.1.1 Problem: Going Beyond Best Effort Service

The original ARPAnet routing algorithm developed by BBN [8], and the precursor to the Open Shortest Path First (OSPF) protocol [69], computed routes based on delay criteria -- queuing

delay being one specific type of criteria. As links became more loaded, delay increased, causing the routing algorithm to compute other routes. Unchecked, this approach was susceptible to routing oscillations. Enhancements to the algorithm, such as basing the metric on router utilization [92], help reduce the level of oscillations.

Today's version of OSPF focuses on a single metric to calculate routes, which is generally based on a shortest path or hop count metric. There has been a proposal in the IETF to augment OSPF so that several metrics can be distributed which reflect varying qualities of service. This has the cause and effect of a router producing distinct and potentially different paths to the same destination.

- **Class of Service**

Until recently, the class of service attributed to multicast data packets, using UDP as the transport protocol, has been based on best effort without explicit flow control. As a result, data packets can be dropped due to congestion, or they can experience large variances in delay; both of which can cause poor performance in the application. Within an experimental environment, where the network is either over provisioned or has source thresholds (as evidenced in the Mbone), this type of framework can suffice for most one-to-many type of applications. But in going beyond today's experimental environment, recent work has focused on designing and defining new classes of services [22, 23, 62, 87, 99]. Within the IETF, the term Integrated Services has been used to encompass two new classes of service for both unicast and multicast data: 1) Control Load, which focuses on providing a minimal level of end-to-end bandwidth, and 2) Guaranteed Service, which provides a minimal bound of end-to-end bandwidth and delay. The design of these services focus on data flows extending from host to destination(s) and does not address aggregation of services at a given edge; be it an administrative or topological boundary [45, 59, 111].

Differentiated Services (diff-serv) is an attempt to provide a simple and more generalized means of providing varying levels of services to data packets. The model for Differentiated Services stems from 3 basic contributions. Firstly, Dave Clark identified the possibility of subscription-style profiles being used to place service-level agreements on a per-user basis into an ingress router of a given network boundary[17,18]. The user can be defined as an application, a host, a site, or a prefix of locations – in other words, any level of aggregation. Subsequent to this,

Nichols, Zhang, and Jacobson have defined a model for a Bandwidth Broker, which is similar to the RSVP Policy Server ideas that are being advanced in the RSVP working group of the IETF [100]. The main differences between the two are Time-scales and Scope. By the former, I mean that diff-serv focuses on subscriptions of services by users instead of time-scales of reservations. And by the term scope, I mean that diff-serv supports aggregation of users as well as several levels of granular service [42, 74].

A second contribution in the evolvement of diff-serv came from Jacobson and Clark in which each introduced separately defined models for service based on financial incentives in relation to the marking of transitive packets[18]. A key assumption in this approach is that deployment would be easier, in comparison to the Integrated Services model, because in-line signaling would be used; e.g., setting existing TOS and reserved bits in IPv4 and precedence bits in IPv6.

Thirdly, several authors [10, 102] have gone into some detail in showing how the queuing model (priority queuing) can be used to deploy several levels of service, provided that the higher quality services are mixed with a healthy quantity of Best Effort traffic. The end result is that if the current shortest path does not meet the requirements of a specific service, then either the user must change its requirement, or use a different path.

4.1.2 New Framework for Broadcast Networks

In discussing changes to the IP service model, I acknowledge that different requested services can lead to different paths from a source to a destination. This difference can range from totally disjoint paths to simply different ingress/egress routers attached to the same broadcast LAN. Since the current framework of multicast relies on filtering incoming packets based on incoming interface, the Designated Router has been defined to ensure that: a) only one node will be used to forward traffic into the broadcast LAN, and b) only one node will be used to forward explicit joins to a shared core. However, the process of electing a Designated Router is an arbitrary one that never allows multiple nodes to act as a parent to a specific group of downstream nodes and/or receivers.

In Figure 9 below, I present an example in which two paths represent distinctly different attributes. In this example, Host-A acts as a receiver for 2 multicast groups: a Video lecture of a

single source sent to group {224.1.2.3}, and a service location announcement sent to group {224.2.3.4}. The Video lecture is a streaming application, such as a television channel, that has only one source and no feedback sent from any of the receivers. Since there is no direct interaction between the source and its receivers, end-to-end delay of data is viewed as low priority characteristic, while high bandwidth of the stream or branch is considered highly desirable.

In contrast to the video data, Host-A also receives periodic data transmissions (e.g., service location announcements) that are relatively infrequent and consume low bandwidth. Given this second type of application and the data it transmits, the receiver uses an ingress node whose branch/path provides low delay, low bandwidth, and low data loss. As a result, Host-A uses a different parent for the two different multicast groups.

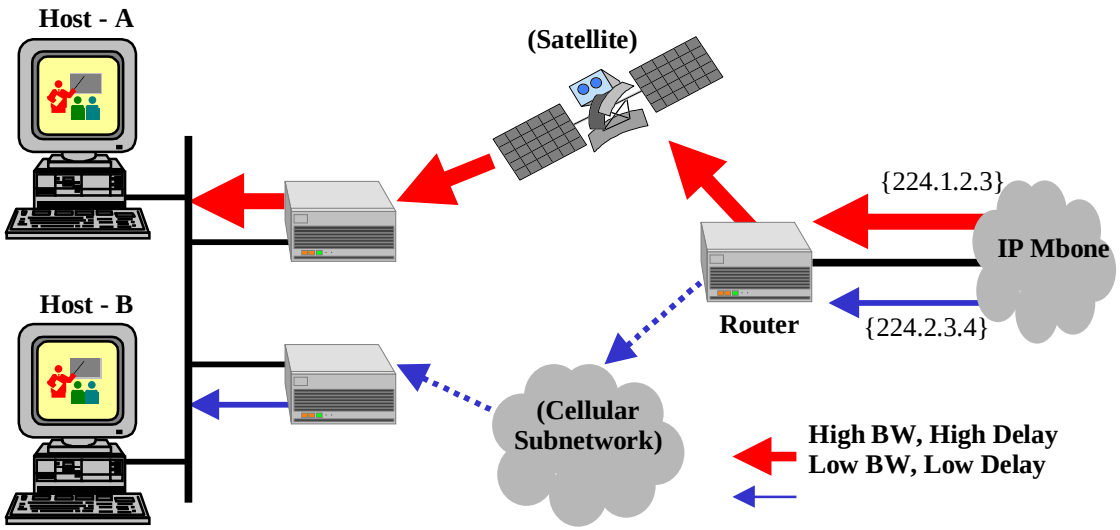


Figure 9: Two groups that have different QoS

Given the effort in introducing new receiver-driven service models, as discussed above, the example of Figure 9 could be expanded so that a different host/receiver (Host-B) chooses a less 'expensive' branch for video traffic that implies a corresponding less granular or lower quality reception of video data. Specifically, this could result in Host-B selecting a different ingress node than Host-B. To accomplish this versatility and variety of requirements, a change has to be made

in the broadcast model of the existing IP multicast framework concerning the election and responsibilities of the Designated Router. In the following, I present a short term and long term solution to the problem of supporting multiple parents on the same broadcast LAN.

- **Short Term Solution: Single-Hop Encapsulation**

One approach in distinguishing different parents on a broadcast LAN is to have the different routers on that LAN establish single-hop encapsulation tunnels amongst each other [28]. By using encapsulation, one can establish virtual point-to-point tunnels to uniquely identify the previous hop of the multicast data, thus preventing loops or a duplication of multicast traffic. However, I view this approach as being short term solution to the problem. This is because the establishment of tunnels is generally an out-of-band process that is separate from routing and is done on a non-real-time basis by administrators.

- **Long Term Solution: Target Router**

I propose a long term solution by augmenting the existing IP multicast framework with the concept of a Target Router (TR). As the name implies, the TR represents a conceptual shift in how the parent node is chosen in that downstream nodes (receivers or on-tree routers) make an active decision and select the node that acts as the parent in a broadcast LAN. The decision can be based on a first-come-first-selected basis, which essentially follows a Best Effort or default selection paradigm. Or, the decision can be based on specific QoS of the downstream branch. This latter example can be realized through specific routing metrics, or exemplified by the in-line signaling provided by the TOS bits in IP packets. In either case, the criteria used to filter downstream traffic is now broadened to include previous hop information. Note that the TR operates within the context of the existing Query/Report model proposed for IGMPv3 [29].

Figure 10 below presents a high-level functional state diagram for a host that operates in the augmented IP multicast framework. The Initial state indicates that the host is not a member of a group. When an application joins a group, the Initial state transitions into one of two states. The Wait state is entered when there is no TR specified, and the filter (for determining the parent) is cleared. In this state, all multicast data for the group is accepted by the receiver so that it can receive traffic from sources directly attached to the shared LAN.

If a Query message is received from a router, then the state transitions to the Steady state, and the filter is set according to the contents of the message. The Query message format differs slightly from that proposed in [29] in that it is augmented to include a list of other active parents for the multicast group. This allows the host to filter out other routers that are acting as parents to different downstream nodes. In response to the Query, the host sends a Report message, which includes the address of the TR. This address indicates that there is still a downstream node associated with a particular parent.

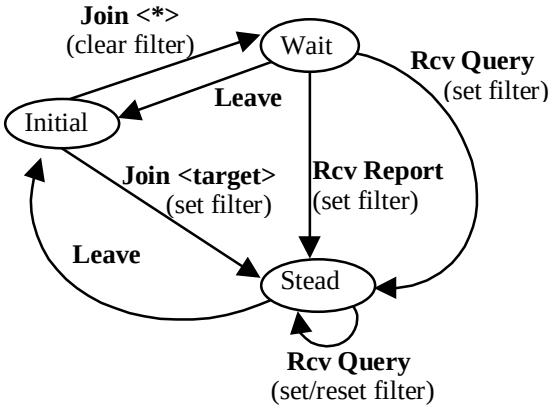


Figure 10: Functional State Diagram for Host

In a departure from the current architecture model of IGMP, two criterion cause upstream routers to send a Report message to the broadcast LAN. One is when a router is grafted onto the distribution tree by the multicast routing protocol. The other criteria is when a host on the broadcast LAN issues a Report (join) message. In both cases, the Report message from the router acts as an advertisement so that other downstream nodes for that group, as well as other parents on the broadcast LAN, learn of its presence. This scheme also allows hosts to only gather state when it joins a group.

- **Related Work: Multicast with Ad-Hoc Networks**

One of the newest areas in research and engineering in IP routing involves the development of routing protocols to support Ad-Hoc networks [19, 20, 48]. The term Ad-Hoc networking describes the case in which routers and hosts move in relation to each other. During this

movement, the routing protocol is expected to forward data from source to destination for both new and existing connections. Initial efforts in this area have primarily focused on providing routing support for unicast traffic only. Multicast support is generally viewed as an area of subsequent research and development, though one group has investigated the use of application level multicast [61].

In relation to the work I present here, Ad-Hoc networking is predominantly outside the scope of my thesis. However, I view the proposed Target Router design and its more granular filter, as being useful, if not critical, to the success of supporting multicast in an Ad-Hoc network environment. This is because by its very nature, changes in Ad-Hoc networks can be expected to alter its topology, potentially producing one or more parents per broadcast area.

4.2 Reducing the Cost of Discovery

The one-to-many join mechanism is the focal point by which the YAM protocol discovers a set of on-tree nodes and potentially receives a number of choices in which to graft a branch onto the shared tree. This join mechanism, as specified in YAM, acts as an on-demand means of discovery that is terminated in one of two ways: a) reaching the edge of the network, or b) reaching an on-tree node. This implies that a significant cost is placed on the system in terms of the amount of control messages that are generated by each new group member.

In Chapter 3, I presented the design of a three-tiered signaling schema for YAM meant to aggregate the number of joins per region or domain. At the edge of a domain, a single node, termed the Joining Node (JN), would act as the proxy that would generate one-to-many joins on behalf of the receivers within its domain. However, if the system has a large number of JNs generating one-to-many joins, then the problem of control message overhead remains a topic of concern amongst the transit domains or nodes. Below I present four approaches that can be used to reduce this potential cost[16].

- **Expanding Ring Search**

Expanding ring searches using IP datagrams involve retransmiting the same message with ever increasing time-to-live (TTL) values. This sequential process allows the message to reach an

[16] Chapter 5 will provide a quantifiable examination of what this cost can be via simulation

ever increasing set of nodes with each transmission – thereby, forming a topological ring emanating from the source of the datagram. When sufficient replies have been received, the search is terminated. The purpose of using this type of search for the YAM protocol would be to constrain the distribution of the discovery message so that only a subset of nodes receive the message.

The drawback to this approach is that the nodes within the ring, i.e., those nodes within a hop count less than or equal to the TTL value, have received the same message (T–H)+1 times, where T is the TTL value and H is the hop count distance of the node from the source. [28, 29] represents the total cost as:

$$\sum_{i}^{t} w(w-1)^{i-1}$$

Where (t) is the maximum TTL value, and (w) is the average degree of the routers in the network. To ensure that an on-tree node is found, the value of (t) is initially set to widest diameter of the network in terms of hop count.

This is an interesting means of characterizing the cost, but in the case of the YAM protocol it is incomplete. The reason I state this is because on-tree YAM routers terminate the one-to-many message, and therefore the diameter of the network decreases as new receivers join the group and install new branches onto the tree. The grafting of new branches effectively partitions the network, and thus reduces the maximum distance in which the discovery message can be distributed. However, while the cost is reduced with an ever increasing receiver set, it can be still considered significant – especially as the initial receivers join the group.

Another concern in using this type of discovery process is the overall delay experienced in obtaining responses from on-tree nodes. With a single unbounded TTL, the delay in obtaining a response is (1) RTT between the source of the discovery message and the on-tree node. In using an incremental series of TTL values, the delay is increased to that of the sum of each RTT for each TTL value.

- **Routing Manager**

The QosMIC protocol splits the discovery mechanism into two processes working in tandem. One involves an expanding ring search with a maximum TTL value. The other process involves a Routing Manager (RM), which acts as a central repository of information for a group and keeps

a list of on-tree nodes. Each router that is grafted or pruned from the tree registers this information with the RM.

When a new QoSMIC leaf router attempts to graft a branch on behalf of a receiver, it sends a limited number of expanding ring discovery messages as well as a query to the MR. The MR then sends a message to a set of on-tree nodes requesting that they send a message to the new leaf router, thus informing it of multiple paths from which to graft a branch onto the tree.

There are several concerns with this approach. One involves the amount of information that must be stored in the MR in conjunction with the potential implosion of signaling information regarding prune/graft information. Another concern involves a potential single-point-of-failure problem attributed to this type of client-server architecture. Finally, one needs to take into consideration the means by which the MR advertises its presence to other nodes in the network and its cost to the system.

- **Caching Discovery Messages**

A non-deterministic manner of discovering on-tree nodes involves caching of one-to-many discovery messages sent by other nodes. In this case I assume that originators of one-to-many joins have themselves joined the group. Thus, by caching this information, a joining node can send unicast discovery messages to these other nodes without sending a one-to-many join itself.

I classify this approach as being non-deterministic because the cached information may be stale with respect to the other node having join and left the group. Since leaving a group does not involve a one-to-many notification, there is no way of explicitly notifying other nodes in changes of the status of a cached entry. The fact that this is a non-deterministic mechanism means that caching of information must be well engineered. Further, the joining node must be able to send a one-to-many join in cases where it receives no positive responses from its unicast discovery messages.

- **Constrained RPF -- Subcasting**

At its most basic level, the term subcasting refers to constraining the distribution of multicast to a subset of all receivers for a given group. However, divergent viewpoints can arise depending on

which layer of the reference model one is focusing on. From the perspective of reliable multicast (RM), subcasting focuses on ensuring that retransmissions of data are targeted towards specific subsections or branches of a tree so that repeated data is bound to a subset of the group. In the case of anypoint distribution trees like OCBT, flags or labels will need to be used to indicate the direction of the distribution. Single-point distribution trees like PIM simply need the identity of the parent node from which the limited distribution starts. However, the means by which this is accomplished, as well as the general subject of RM, is beyond the scope of my thesis.

From the perspective of routing, subcasting focuses on the manner in which the whole of the potential multicast tree is constrained within a topological boundary. One of the more common ways of achieving this is through scoping: by administrative designation of addresses, or through the use of TTL [67]. Beyond this, recent work [12, 13] has focused on investigating different ways in which multicast distribution is topologically bounded between a source and a specific unicast destination. An example of the latter case is presented below, in which a type of subcasting is used to discover alternate paths, as well as on-tree nodes, that may exist between the source and a given destination unicast address.

Directed Spanning Join

A derivation of the one-to-many join mechanism, within the context of subcasting, is referred to as a Directed Spanning Join (DSJ). Its goal is to discover multiple paths to a given destination in addition to reducing the impact of source initiated reverse path forwarding.

The distribution of DSJ messages is similar to that of RPF, in that the message received from the incoming interface is sent to the other interfaces **if** the incoming interface is the shortest distance to the source of the message. Beyond this rule, the DSJ control message includes a unicast destination address and a routing metric value. The former item identifies the ultimate consumer of the join message – i.e., the target node. The latter item identifies the current distance of a previous hop to the destination address. If the routing metric of the previous hop is greater than or equal to the metric of the current node, then the metric value is updated and the message is distributed via DSJ. However, if the metric of the current node is greater than the metric from the previous hop (indicating that the destination unicast address is "further" away), then the message is dropped.

Figure 11 below shows an example of the distribution of a directed spanning join message. In this example, a source-node (one that initiates the DSJ) broadcasts the join message through all of its interfaces. The nodes that are "farther away", in terms of number of hops, from the unicast destination or target node terminate the message and discontinue the distribution. Other nodes that are of equal or less distance continue the distribution of the DSJ message. As a result, only the nodes in the shaded region of Figure 11 receive the directed spanning join message. Other nodes that do not conform to the rules of the DSJ message do not receive the discovery message.

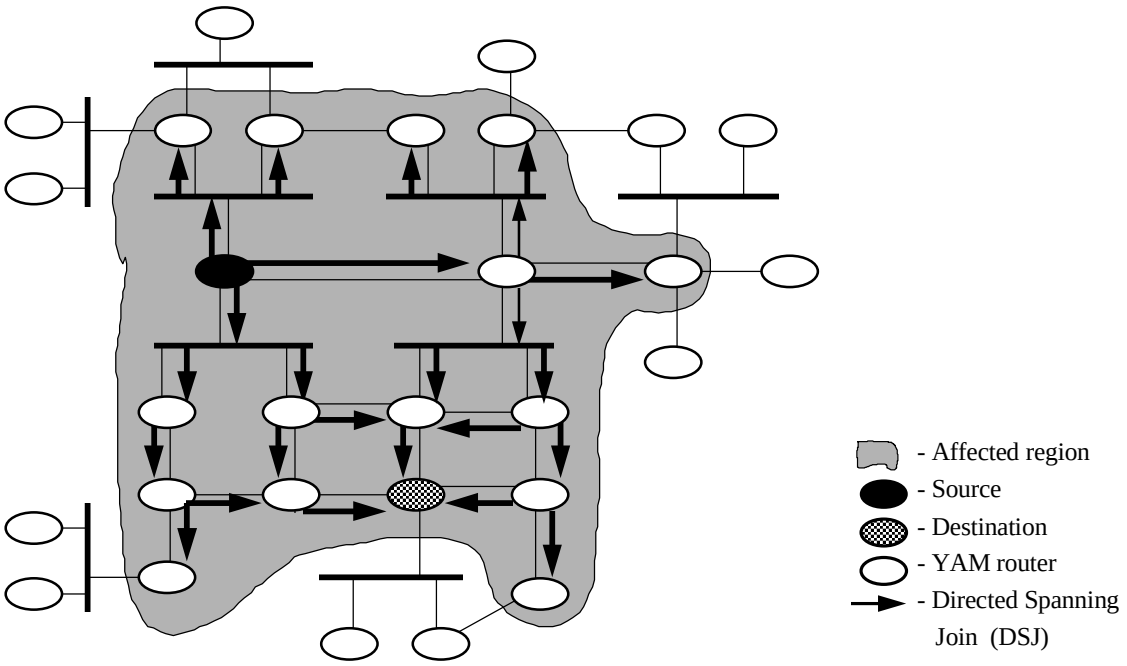


Figure 11 - Example of Directed Spanning Join

The value in using DSJ to propagate the discovery message is realized by having a constrained number of nodes be burdened with explicit join messages. And while the above example takes the perspective of a joining node sending a discovery message towards an on-tree node (one example being the core), I could also apply its discovery property from the perspective of an on-tree nodes towards a leaf router. The limiting component of this strategy, in comparison to the unconstrained one-to-many join, is that it is dependent on the identification of a target node.

4.3 Some-to-Many Communication

Some-to-Many communication represents a communication model in which a subset of nodes act as sources to the group through some period of a session. This subset represents more than one source, but less than all possible sources or all members of the group – the latter example constituting *many-to-many* communication and exemplified by Distributed Interactive Simulation (DIS)-type applications. The problem in characterizing the *some-to-many* model is that the characteristics of a given source can change dramatically in comparison to other sources. For example, one source may be active during the entire session, another may be active for just one period of time, and yet another source may be active on a periodic basis, and then become dormant. In addition, there is no definitive means of determining how many nodes will act as sources. Detailed examinations of application behavior, including that of some-to-many communication, are outside the scope of my thesis. However, the issue remains as to how a protocol such as YAM could begin to address several and potentially intermittent sources.

In Chapter 3, I presented 5 examples of metrics or criterion that can be used to construct a tree. Three of the metrics used the core as the point of reference in determining the best path to graft a new branch onto a tree. Given this single reference point, it stands to reason that measurements obtained from other on-tree nodes acting as sources, would be worse than the same measurements taken from the core. Conversely, other metrics that do not use the core as a reference point could be expected to show better results in terms of average end-to-end delay. In Chapter 5 and 6, I present simulation results that strengthen these assumptions.

If, however, the objective in addressing the *some-to-many* scenario involves achieving a near-optimal solution space for a given metric, then three basic approaches can be taken in evolving the YAM design presented in Chapter 3. The most radical approach involves altering the design so that YAM becomes a Hybrid type of tree construction that builds a shared tree as well as source tree for each additional source. While this approach would produce the best results for each new source, it would also produce **<s,g>** state per source and run contrary to the goals of generating a shared tree – reducing the amount of stored state in the network. A less radical approach would involve the migration of the core to reflect a more optimal placement with respect to existing sources sending traffic to the bi-directional tree. However, as I discuss in section 4.5 below, this option also has its drawbacks in the form of packet loss and/or duplication,

which in turn can become an acute problem if location and duration of sources changes at a rapid pace. A third approach involves on-tree nodes keeping track of a set of sources to the group. To reduce the amount of information that is stored, the set of known sources can be configured to have a maximum limit of sources (e.g., 10). Precedence can be defined in terms of the order in which sources have been tracked (last known source is labeled as first on the list), or the amount of traffic attributed to each source. Then, when a one-to-many join is received, the on-tree node is sent the list of sources and the corresponding metric value representing the cost of the path between the on-tree node and each source. Once the information is received, the leaf router can then decide which on-tree node is used to graft a new branch onto the tree.

Without having a more definitive set of characteristics to attribute to a *some-to-many* scenario, one cannot provide a definitive answer as to which of the above tree approaches should be used to continue the evolution of YAM. However, given the interesting simulation results of Chapters 5 and 6 showing the results of the Fan-Out metric and FCFS criteria, it can be argued that existing non-core-centric metrics may be best at addressing the *some-to-many* problem space. This is in addition to the previously stated understanding that a shared tree will never produce as good a set of results as a source-tree-per-source of the group.

4.4 Shadow Prices

In [57], Kelly presents the concept of Shadow Pricing and its integration with rate control algorithms in order to achieve stable convergence in establishing a per unit charge of resource used to forward traffic. The cost associated with a resource is in relation to the other flows sharing the same resource. As the demand increases, either by new flows or added demand by existing flows, the cost rises. By increasing the cost in relation to an increase of demand, the system provides an equalizing influence between the diametric forces of supply and demand. The cost decreases as less resources are requested, while the total cost incurred by the source is the additive sum of the cost of each resource along the path to a destination.

Kelly conceptualizes a model in which the request for resources induces a corresponding feedback in the change of the shadow price to all users of the resource. Rate control is then used by the elastic application to respond to the current price. The foundation of this work is based on original work by the 18'th century mathematician Lagrange, who developed the Lagrange

Relaxation and Multiplier techniques for nonlinear optimization. Whittle [107] expands on the notion of shadow prices within the context of using constraints in attempting to achieve optimization.

In general terms, the Relaxation and Multiplier techniques focus on combining non-related metrics into a single modified metric. The relaxation involves the strategy of removing complicated constraints (e.g., multiple metrics) and replacing them with a function associated with multipliers. The multiplier serves as a common weight that is used to alter the characteristics of one metric so that it can be combined (e.g., added) to a different metric. So for example, if one metric related to cost, and another metric referred to delay, then the multiplier can be applied to the time metric to produce a modified cost metric. The two values can then be combined to form a single cost metric, upon which a shortest path route can be computed in polynomial time.

In the case of Kelly's model, shadow prices act as the Lagrangian Multiplier. Further, its usage in an economic model together with feedback in changes of the shadow price play a role in influencing the rate at which traffic is injected into the network by the source. However, what are left as open issues are a) the user's reaction to the rates allocated to them by the network, and b) the means by which feedback is effectively propagated back towards the source in an IP based network. One can add to this by inquiring as to other actions that can be taken in reaction to changes in shadow prices. In particular, what would be the impact of alternate routing given equal-cost multi-paths, or unequal cost alternate paths that can satisfy the Min-Max fairness criterion[30]?. I qualify the above with the understanding that Kelly's work assumes a network that forwards unicast flows between source and destination.

- **Shadow Prices and YAM**

One of the challenges of YAM involves the selection of the 'best' path used to graft a new branch onto the shared tree. The definition of 'best' is somewhat dependent on the type of metric used to construct a tree. In Chapter 3, several different types of metrics or criterion were defined, and later in Chapters 5 and 6, an examination of these metrics in relation to each other is presented through simulation. In defining these metrics, I also place constraints or qualifications that determine the quality of one path over another. However, these qualifications are predominantly focused on the atomic characteristics of the path itself (e.g., lowest delay) – be it just from the on-

tree node to the Joining node, or from the shared core to the Joining Node. The issue remains in terms of selecting a path based on the resource(s) used by the new branch. These resources can involve buffer space, queue management/prioritization, storage of state, etc.

The integration of shadow prices with the design of the YAM protocol would provide an economically based schema that can refine the decision process in selecting a path as a new branch on the tree. The Query-Reply design of YAM provides a commonality that allows the two disjoint mechanisms to operate in tandem. Further, another Type-Length-Value sub-message could be attached to the YAM Information message so that other downstream nodes are informed when shadow prices change. In order to avoid oscillations, thresholds could be defined and updated gradually that would limit the number of Information message sent to downstream nodes. Chapter 7 provides additional aspects that could be investigated regarding the design of YAM and shadow prices

4.5 Core Migration

The baseline design of YAM focuses on the construction and maintenance of a bi-directional shared tree whose branches are constructed based on a particular type of metric or criteria. The protocol primitives of YAM have been defined to allow the inclusion of a variety of non-critical augmentations to its initial design. By non-critical, I mean functionality that can be omitted with no disruption in the atomic construction and connectivity maintenance of the tree.

One example of a non-critical function is core migration. Within the context of changes in group membership, this function attempts to change the designation of the core so that it reflects a better location. What is considered 'better' is dependent on what is viewed as pertinent criteria. In [31], the authors use the topological distance between all receivers and the core versus other off-tree nodes and their distance to the receivers as a means of deciding whether the core should migrate. If the off-tree node displays a better average hop count between itself and the receivers, then those receivers tear down their branches and graft new ones onto the newly elected core.

While such a determination may seem initially advantageous, consequences of such actions need to be weighed before any arbitrary relocation is incorporated in a shared tree routing protocol. For example, if downstream nodes are unilaterally instructed to prune their upstream branches and graft a branch onto a different node (i.e., a new core), then it can be assumed that some

measure of in-transit data will either be duplicated or lost. Given an environment of Best Effort service and IP multicast operating over UDP, packet loss is an accepted condition, although it is certainly not to be encouraged or promoted in a frequent manner. On the other hand, if the tree conforms to a particular QoS and the resources along the branches have been reserved accordingly, then packet loss (not due to connectivity failure) becomes an acute issue to be avoided – presumably, regardless of the gains of core migration.

The issue of packet duplication is perhaps a more intractable problem in weighing the benefits of migration as outlined in [31]. I say this because while packet loss is a known quality of shared data networks, duplication is an anomaly that is commonly attributed to a form of denial-or-service attack. In cases like the Network Time Protocol [112], such attacks cause clients to ignore or discard information from a previously 'trusted' source. Again, the perceived benefit of core migration complicates the perceived benefits of core migration. Thus, given all these factors, and the associated cause and effect they produce, I term approaches like that of [31] as *complex* migration strategies.

A different approach in core migration focuses exclusively with on-tree nodes and is based on the number of downstream branches emanating from the shared core. The rules for migrating a core entail the following:

1. If the Core has no receiver within its Region, and there is only one downstream Child attached to it, then send a downstream *Elect-Core* message and remove state for the group.
2. If the downstream node has one Child, then continue forwarding the *Elect-Core* message downstream and remove state for the group. However, if the node has more than one child, then terminate the *Elect-Core* message and elect itself as the core for the tree.

Figure 12 below presents an abstract example of the above rule set. In Case-1, a receiver leaves the group causing a prune action that removes the branch and receiver from the tree. The Core then sends an Elect-Core message downstream, which is forwarded until it reaches Node-A. Since Node-A has two downstream children, it terminates the message and elects itself as the Core for the group.

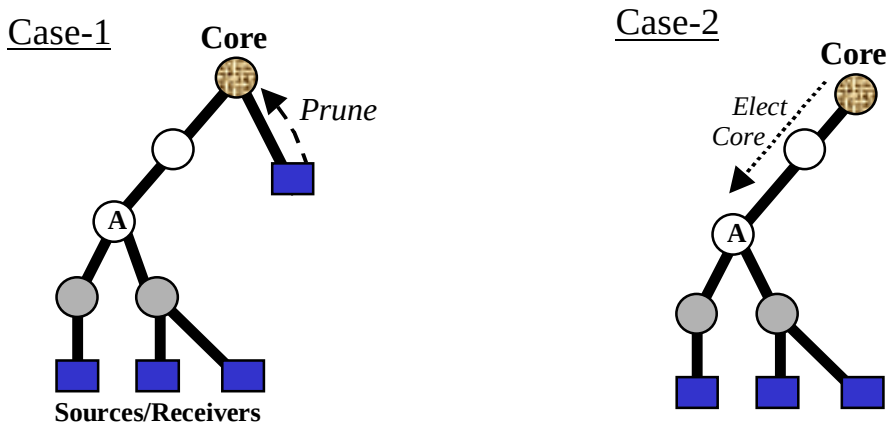


Figure 12: Example of Simple Core Migration

Part of the value in this action is that by moving the core closer to the existing receiver set, one eliminates state and data in nodes that have no use for it (i.e., the branch segment between Node-A and the previous Core). More important, in comparison to the Complex migration strategy outlined in [31], is that no data is lost or duplicated in this schema. Given the minimal rule set and cost to the system, I term the above as a Simple core migration strategy. A simulated exercise of this strategy is presented in Chapter 6. One aspect to keep in mind is that the Simple migration strategy only takes into effect when there is a single downstream node from the core. Hence, its usage may only occur in an infrequent basis (if at all) during the lifetime of a groups existence.

4.6 Summary

This chapter has presented several issues that arise in supporting QoS multicast within a broadcast LAN environment. I have shown that the existing practice of arbitrarily selecting a designated router to forward all data and control messages is incongruent when different paths are used to satisfy different QoS. To address this issue, I presented an approach that allows different nodes in a broadcast LAN to act as a parent for a specific set of downstream nodes. I refer to these parents as Target Routers (TR).

With a TR, receivers and downstream nodes can use local criteria to choose the specific node that acts as its parent, as opposed to relying on the input interface as the sole filter for distributing incoming data. The choice of the TR can be based on existing load, in an attempt to perform rudimentary load sharing, or on some other more granular characteristic (such as delay, loss, etc). This added versatility in choosing different parents can be particularly helpful in wireless environments, where other factors like signal strength and available transmission slots play a more significant role than that found in traditional wired systems. Outside of the scope of routing protocols, the TR concept allows multicast protocols in broadcast LANs to support the same administrative policies used in unicast routing. This is because the selection of the TR is based on a combination of the incoming interface with the IP address of the parent.

In this chapter, I also looked at a number of strategies that can be used to help constrain the impact in using a one-to-many join mechanism to build a shared tree. One strategy discusses the use of a proxy node, termed the Manager Router, to trigger messages from on-tree nodes to a leaf node attempting to graft a branch onto the tree. From these messages, the leaf-router selects the path that best matches its target metric. I also discussed the use of caching discovery messages and using expanding-ring searches, as opposed to a single unbounded one-to-many discovery message, to discover alternate paths towards the tree. Thirdly, I presented an alteration to the Reverse Path Forwarding algorithm to include a metric from the source to a specific destination. As the message is propagated from the source, neighbor nodes use the metric to determine if the discovery message is to be forwarded or terminated. I titled this augmentation of RPF as a Directed Spanning Join.

Regarding the field of network economics, I discuss the issue of refining the process of selecting paths by using shadow prices as an additional discriminating factor. This new schema introduces a more economically influenced decision process that has the potential to achieve an equilibrium of network supply and receiver demand of resources. This in turn would produce yet another discriminating factor in the selection of a path as a new branch on the shared tree.

Finally, I discussed the topic of core migration. Two different paradigms were presented, Simple and Complex, with the pros and cons of each being presented.

Chapter 5

Examining the *Construction* of Shared Trees via Simulation

Introduction

In Chapter 3, I presented the YAM protocol, which is designed to produce QoS sensitive shared trees. In this chapter I examine the protocol through simulation to gather information concerning the impact of its tree construction mechanism as well as to compare the different types of trees that can be constructed using different metrics or criterion.

Within the context of IP multicast, a primary purpose for shared trees is to reduce the amount of state stored in a network for a multicast group. The current set of shared tree routing protocols, such as Core Based Tree (CBT), Protocol Independent Multicast (PIM), and Border Gateway Multicast Protocol (BGMP), build the tree over a unicast routing protocol using a single metric to calculate a single path between a source and destination [53, 69, 72, 81]. Specifically, the source and destination correspond to the leaf router and the shared root; the latter having been selected a priori and irrespective of group membership.

One consequence of using a single metric to construct the tree is that the branches may not reflect the desired qualities of the multicast application. If the application requires a certain type of metric (e.g., high bandwidth), there is no way of assuring that the minimal hop count will effectively satisfy this or some other metric.

Another consequence attributed to today's shortest-hop to the core approach is that the resulting tree tends to resemble a star as opposed to hub-spoke topology. On initial inspection, this would seem to be desirable because the former would be expected to store state into a minimal set of

routers in the network. However, as one shall see in this paper, star topologies tend to promote higher fan-out (a large number of downstream interfaces) near the shared root or core, while hub-spoke topologies tend to promote high fan-out towards the edges of the tree.

The primary focus of the following involves examining YAM in terms of different metrics and its cost to the system. Additional comparisons are made from a variety of perspectives and conditions that can be attributed to other types of IP multicast protocols – examples being single source trees, and arbitrarily selected cores. To provide a point of reference, I identify these attributes with a specific protocol such as CBT or Express. However, it is important to note that this latter comparison does not focus on protocols per se, but the resultant trees and/or data distribution that each protocol would produce.

5.1 Design of YAM Protocol

As stated in Chapter 3, the YAM protocol is designed to construct trees using one of several different types of criterion or metrics. There are currently five different types of metrics that have been defined for use by YAM. These metrics are:

1. Hops
2. Delay
3. Bandwidth
4. Fan-Out
5. First-Come-First-Served (FCFS)

The Hops metric produces the same tree as that produced by protocols like CBT or PIM-sm. The response that has the lowest number of hops between the leaf router and the core is the one that is selected by the leaf router. The Delay and Bandwidth metrics produce responses that reflect the paths that have the minimal amount of delay and the maximum amount of available bandwidth from the core towards the leaf router, respectively.

The last two metrics reflect a different approach in defining criteria for constructing a tree. The Fan-Out metric is designed to promote high fan-out near receivers. It accomplishes this by selecting the response from an on-tree node that is topologically close and has a high number of

child nodes attached to it. The FCFS metric, or rather criteria, is the simplest of the previous four and selects the first response it receives from the one-to-many join. I shall show below that this last metric produces trees that are almost always similar in structure to the Fan-Out metric.

5.2 Related Simulation Work

Within the context of simulation, [9] compares and analyses some of the differences between early versions of CBT and PIM protocols. The analysis focused on state and delay between a CBT-based tree and the different types of trees (shared or hybrid) that can be formed by PIM. Since both of these protocols construct the tree using shortest path/hop metric, other metrics or existing load was not taken into consideration. In addition, the simulations used a variety of locations for the shared core in order to obtain a worst/best case examination of its impact in the construction of the tree. Since the publication of the analysis, both protocols have evolved in their design. However, most of architectural design of each protocol has stayed the same. Thus, the case where only shared trees are constructed (i.e., no **<s,g>** state is introduced into the network), CBT will perform as good or better than PIM in terms of average delay from a source to all receivers. This is because CBT uses bi-directional trees whereas PIM uses a single point forwarding mechanism that requires all data be initially sent to the core for distribution.

Recently, [38] presented the design and analysis of the Quality of Service Multicast Internet protocol (QoSMIC). This protocol is a follow-on to an earlier version of YAM and introduces the concept of a Manager Router (MR). This entity acts as an intermediary node which knows the current location of the tree. When a new node attempts to join a tree, it generates a local search as well as sends a query to the MR. Upon reception of the query, the MR informs nodes on the tree to advertise itself to the node that wishes to join the tree. As in the case of YAM, a selection is made from the advertisements based on a specific criteria.

The analysis in [38] is based on a scenario that has only one source for the group during a simulated run. The notion of QoS and congestion is abstracted and correlated to the cost-metric[17] associated with the edges of a graph. If the cost is high, then the link can be interpreted as experiencing congestion or having a low QoS ability.

[17] Cost does not refer to a monetary cost in sending or receiving packets.

Outside the scope of previous simulation work, [62, 99] present approaches aimed at Quality of Service routing in unicast networks. Both of these approaches deal with the distribution of different metrics for each arc of a directed graph. In [99], one is shown that the selection of a unicast path using multiple unrelated metrics is a Non-Deterministic Polynomial (NP) Complete exercise. [62] shows how this problem can be altered to a Polynomial problem space if the network service disciplines are rate proportional.

YAM operates on top of any unicast routing protocol. The selection process used in grafting a branch onto the tree is made by the YAM leaf router basing its decision on the collection of end-to-end paths from on-tree nodes to itself. This reduces the problem space so that the underlying unicast routing protocol is not required to collect all attributes (existing metrics) for all arcs of the graph.

5.3 Simulation Approach

The approach of the simulation effort presented in this chapter is to examine and compare the types of trees that are constructed with different metrics. This chapter does not duplicate the work discussed above in Chapter 4, but rather will advance this work to provide additional insight in constructing shared trees using different metrics. Unlike [38], this paper will quantify specific metrics and compare them against one-to-many and many-to-many communications amongst the group members. The following information will be used to measure quality and impact of a given type of tree:

1. State
2. Responses
3. Control Message Cost
4. End-to-End Delay

The first type of measurement involves measuring how many nodes in the network maintain state for a particular type of tree. A metric that needs less nodes to construct a tree, in comparison to other metrics, is considered desirable and in line with the underlying goal of a shared tree – to minimize state stored in the network.

The second type of measurement involves the number of responses received by a YAM node that issues a one-to-many join message used to discover the tree. These responses indicates the number of choices made available to a node in satisfying a target Quality of Service.

The third measurement is based on the number of nodes that receive a one-to-many join message for each receiver that joins the group. This number indicates the cost that each receiver incurs on the network as it joins the group.

The fourth and final measurement involves the end-to-end delay of data from a source to its receivers using two primary perspectives: one-to-many and many-to-many. The former case correlates to a single-source model in which only a single node is used to send data to the group (e.g., television over the Internet). In my simulations, I use the core as the reference point for gathering information from the different receivers. The selection of the core is made because of the assumption that the location of the source is also the location of the root of the tree. As an example, if one were to 'view' a space shuttle session over the Mbone, a trace of the multicast tree, as well as its corresponding advertisement on SDR, shows that both the root and the advertisement are located at NASA (the source of the transmission). Observations of other single source transmissions and advertisements on the Mbone show this to be the standard case, thus reinforcing the notion that using the core as a data point in examining different metrics to build a tree is a valid assumption to make.

In terms of many-to-many communication, I treat all receivers as a source to the group. This scenario represents what could be considered an extreme variation of communication, but also it produces definitive results that are not based on probable assumptions or subjective views.

A secondary set of results, presented in section 5.4.1, are taken from three non-core receivers: namely, the 6'th, 11'th receiver, and the 18'th receiver that joins the group. These results are used to display different perspectives other than that from the core in the acknowledgement that certain sessions may have more than one source, and yet not be considered many-to-many communication. However, this additional perspective is to be considered subjective because there are too many open issues to consider (how many other sources should one consider?, to what extent do they act as a primary source in the context of the session's lifetime?, etc.). In addition, it is important to recall that given a bi-directional shared tree, there will always be non-optimal performance from some set of receivers that act as sources.

A specific non-goal of the simulation exercise is the examination of trees based on traffic load. Initially, this would seem to be an interesting characteristic to investigate. But the problem with this type of test is that an 'examiner' can bias the results by selectively placing traffic load on specific on-tree links, thereby selectively penalizing specific types of criterion (e.g., Fewest Hops vs. Shortest Delay). The achievement of such a test only reinforces the weakness of a specific criteria -- how bad it can get. It does not add to the insight and impact of the criteria itself.

5.3.1 Simulation Environment

The simulator used in this effort was version 2.1b3 of the UCB/LBNL/VINT Network Simulator known as NS[18]. The YAM protocol, with its Discovery, Response, and Selection primitives, was added to NS in order to build shared trees based on different metrics.

The graphs shown in section 5.4 below represent one sampling of 40 simulated runs. Each graph from the sampling uses the same number of randomly chosen receivers, the same start time in which receivers join the group, and the same topology and connectivity consisting of 120 nodes. An abstract depiction of the topology, without multicast routes superimposed is shown in Figure 13 below. The shaded region denotes transit-only nodes – routers that are never receivers or members of a group. The unshaded region towards the outlying boundary of the topology represent the set of nodes from which receivers are selected.

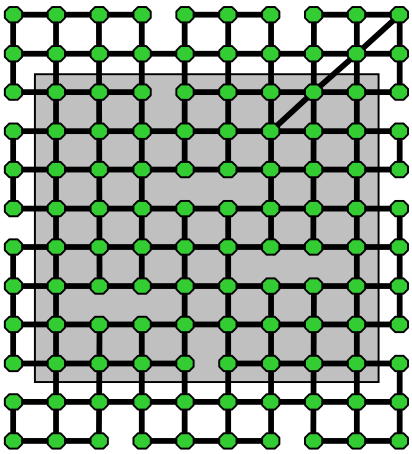


Figure 13: Simulation Topology

[18] Additional information on the NS simulator can be found at: <http://www-mash.cs.berkeley.edu/ns>

The topology represents a partial mesh structure that has a number of links removed in order to increase the chance by which some paths spanning different points along the topological boundaries need to traverse more nodes in comparison to a full-mesh topology.

The purpose of this topological structure is to provide an abstract representation of an internet. Key features in the representation involve: a) locality of connectivity, b) a depth in terms of hop count between some set of receivers, and c) placing receivers/sources at the boundaries of the topology. The rationale behind this abstraction is based on two perspectives. The first comes from work by Ellen Zegura et. al. in [109], which describes a comparison of models and graph structures used to represent an internet topology. In this work, Zegura points out the emphasis of local connectivity between nodes versus a random connection of nodes. In addition, she also emphasizes the transit-stub structure in which stub domains act exclusively as sources or sinks of data and do not act as transit or relaying points of a network. This latter aspect effectively places stubs at the edges of a topology.

A real-life example of this stub-transit nature of the Internet is shown in Figure 14 below, which depicts a domain level view of the connectivity between four stub domains located in different continents (Europe and North America). In this figure, the UCL and Imperial College stub domains are connected to the U.K. Academic & Research Network (JAnet), while the SAIC and UCLA stub domains are connected to the Cable & Wireless transit domain (i.e., an Internet Service Provider {ISP}). Hence, one is shown a snapshot of an actual hierarchy and 'edge' connectivity between transit and stub domains.

In taking this perspective one step further, the traceroute between SAIC and UCL shows that 14 hops, existing in transit domains, separate the two stub domains – reinforcing the need to represent depth in a simulated topology. As a sidenote: two transit hops through Janet separate UCL and Imperial, while 7 transit hops through Cable&Wireless separate SAIC and UCLA.

From a simulation perspective, Doar uses a hierarchy of networks as a target archetype from which to construct sample topological models used to examine routing protocols[15]. Specifically, he focuses on individual LAN's connected to WAN's, which in turn are connected to other WAN's.

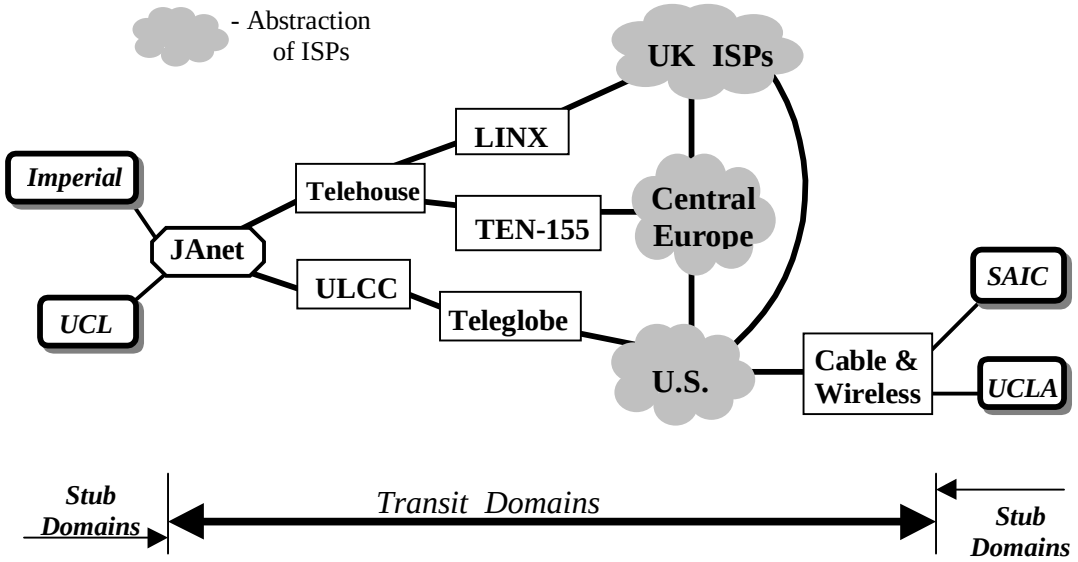


Figure 14: Example of Inter-domain Connectivity

Regarding the simulated topology of figure 13, a certain amount of randomness is introduced. Specifically, bandwidth and propagation delay of each link is randomly selected within a range of 1.1Mb-3.0Mb, and 8ms-14ms respectively. The range of these values are purposely selected to be within a relatively narrow band so that the cumulative length of a **path** is the deciding factor in its selection as a new branch on the tree. In the case of the delay metric, this involves a cumulative sum, while the bandwidth metric (as applied in the simulations) involves the minimal maximum value of the end-to-end path. In other words, the highest available bandwidth throughout the path. Note that the bandwidth/delay attribute of each link is constant for each of the 40 simulated runs, hence, only the selection or receivers changes from one simulated run to another.

Concerning the selection of group membership, 40 sets of randomly selected receivers are chosen from the edges of the topology – one set for each simulated run. Selecting from the edges ensures that a minimal amount of depth (at least 8 hops) will be traversed by some subset of receivers. This ' edge' perspective allows YAM to be exercised in an internet-like topology, where receivers are located at stub domains. The boundaries of what constitutes an ' edge' include the top and bottom 2 rows as well as the single left and right most columns of the simulated topology (i.e.,

the unshaded region of Figure 13). Thus, there are a total number of 56 candidate nodes at the ' edge' of the topology, of which 26 are randomly chosen as members of a group. In each of the figures of section 5.4, the identity of the nodes are in ascending order starting from the top right (0), to the top left (9), to the bottom right (110), and finally to the bottom left (119).

The election of a shared core by YAM is based on the first node that joins the group. In the case of Figures 15 through 19, the shared core is node-2. So in the case of the delay/bandwidth/hop metrics, all receivers will use node-2 as the target reference point for grafting a branch onto the tree. One aspect that is outside the scope of this simulation environment is a change in group membership during a simulated run. By this I mean the case in which receivers join and leave a group, thus potentially changing the formation of the tree. While this subject is considered important, this chapter is concerned with presenting a baseline analysis of the initial formation of the tree.

Each of the 40 runs were exercised five times using a different metric to construct a tree. The mean of the cumulative result of all runs (i.e., 40 sets of values for each of the 5 metrics) were used as the basis for the results presented in section 4. The 95% confidence interval was within 5% of the plotted data.

Note that the NS simulator does not support the IGMP protocol. This is a reflection of the measure of abstraction inherent in most simulators. As a consequence, the receiver and the Leaf Router are treated by NS as a single entity. With respect to the YAM protocol, I continue this abstraction and incorporate the functions of a Joining Node, which sends the one-to-many join, within the same entity that contains a receiver.

5.3.2 Tree Comparison

The comparison of trees is divided into two segments. The first is presented in section 5.4 and compares the trees constructed by three different shared tree protocols: YAM, CBT, and a single source tree protocol – the last one loosely resembling the Express design [56]. In making the comparison of the first two protocols, I bring forth an initial comparison between a design that chooses its core based on group membership (e.g., YAM) and a design whose core is randomly chosen and is not a member of the group (i.e., CBT). An additional defining aspect in this

comparison is that the group membership is considered sparse. This comparison is then brought up again in the second simulation segment in which group membership is treated as densely populated (a receiver set constrained to one area of the topology).

The information obtained from YAM and CBT type trees are based on data collected from each receiver that in turn acts as a source of the group. This information correlates to the many-to-many communication model (described in section 2.5). At another extreme is the single-source model in which only a single source is used to send data to the group (e.g., television over the Internet). I correlate this type of tree with one built by protocols like Express.

The second examination of simulations, shown in section 5.7, looks at a worst case scenario that compares YAM trees against CBT trees that have suboptimally placed cores. In addition, both of these types of shared trees are compared against a source based tree, as would be generated by DVMRP [101]. The comparison uses average delay as the criteria for determining the quality of one type of tree versus another.

5.4 Results

Figure 15 and 16 are actual snapshots of the NAM tool displaying trees constructed by a single simulated run using the Delay and FanOut metric, respectively. An inherent capability of the tool involves a direct correlation between the length of a link and its associated delay. For graphs comprised of a large number of nodes, this correlation can present a distorted view to the point where it can be difficult or impossible to distinguish the topology of a graph. This distortion is exemplified in figures 15 and 16, where the orange lines represent links of the tree, and the black lines represent off-tree links.

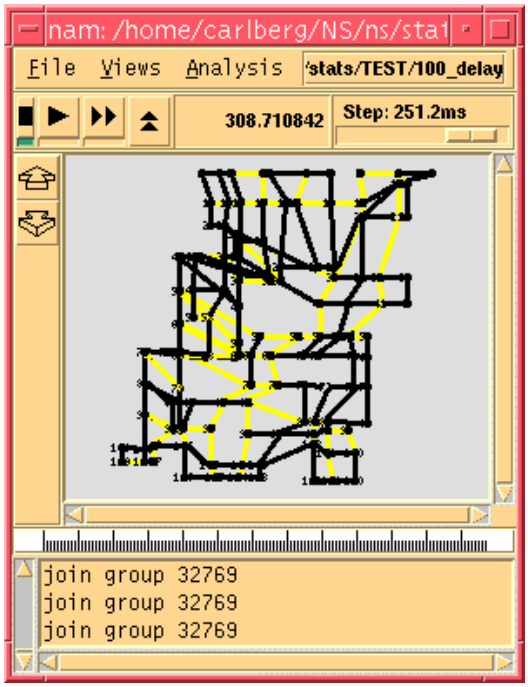


Figure 15: Delay Metric

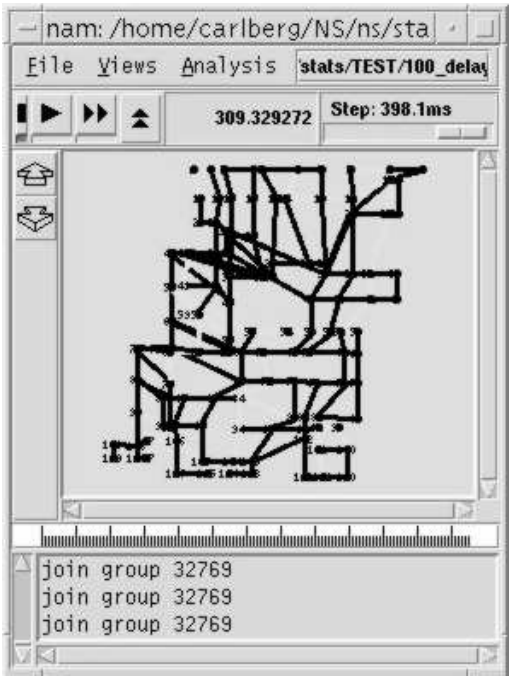


Figure 16: FanOut Meric

In order to present a more visually-perceptable distinction of the topology, and the trees constructed with different metrics, I removed the code in the NAM tool that correlates link length with delay values and instead used a constant value for link length. I also created a series of figures that correlate to the output of the NAM tool. This allows me to more easily distinguish the links that comprise the tree from those that do not.

Figures 17 through 20 represent one simulated run in the construction of a shared tree using different metrics. As mentioned previously, the same core (in this case, node-2) and the same set of receivers joining the group at the same sequential time, are used for all 5 exercises of the simulated run.

In each figure, the tree is shown as thick lines, while the normal (or off-tree) links of the network that have no state for the group are shown as thin lines. Nearly every metric produces a slightly different tree. The exception to this statement is that both the Fan-Out and the FCFS metrics

build the same topological tree (Figure 19). The duplication of the same topological tree occurred in 39 of 40 simulation runs, even though the definition of each metric or criteria is syntactically different.

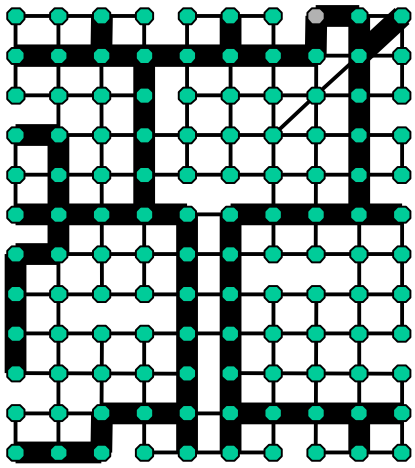


Figure 17: Bandwidth

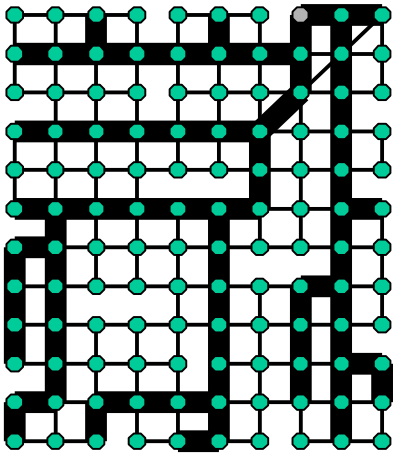


Figure 18: Delay

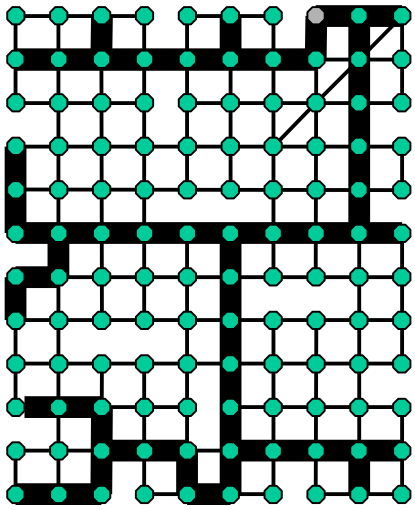


Figure 19: Fan-Out & First-Come-First-Served

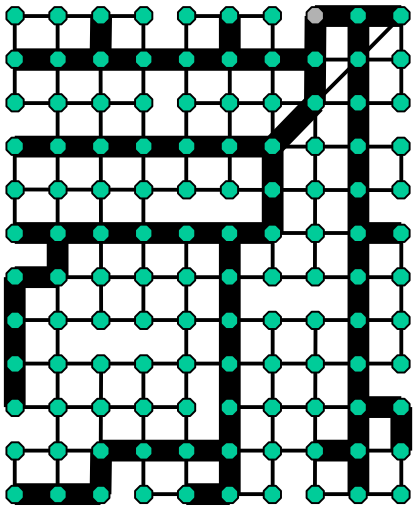


Figure 20: Hops

After the tree is constructed, each receiver, including the shared core, sends a single datagram down the tree to measure the end-to-end delay from that group member. Each tree is bi-directional, meaning that the distribution of the data starts at the location of each source. This is

in contrast to the PIM shared tree model in which data at the source is first encapsulated and sent to the shared core or rendezvous point (RP), which then decapsulates the data and sends it down the tree towards the group members.

The transmission of each packet is accomplished once, and also when no other sources are sending data through the tree. This insures that the measurement reflects the characteristics of each branch, as opposed to load concentrations from other sources. Since, load is not a factor in these tests, aspects like packet loss and burstyness are not tracked by the simulation.

The data that is collected in each simulation is presented in four tables. The first three tables contain information relating to trees that would be constructed by the YAM, CBT, and Express protocols, respectively. This representation was accomplished by including or excluding specific segments of information. So for example, a YAM tree built using a shortest-hop metric, and the exclusion of the core as a source of test data, resembles the same tree constructed by today's CBT protocol. It is important to note that these tables do not represent comparisons of specific protocols, but rather comparisons of the trees produced by these protocols.

Each of the first 3 tables contains three sets of information relating to the mean end-to-end delay of data (in terms of seconds) sent through the different trees. The first column represents the metric used to construct the tree. The second column shows the *Highest Mean Delay* from the set of all sources to all receivers. By this I refer to mean of the highest delay of a <source, receiver> pair of all 40 runs. The third column shows the *Average Delay* of all sources to all receivers for all 40 runs

Both the second and third columns are also accompanied by the difference in percentage of each metric for the given table, thus providing a more direct comparison between the different types of trees. The metric whose difference value is 0% represents the 'best' delay value for that simulated run.

Table 1 shows results of trees constructed by YAM. Each tree is bi-directional and all group members act as senders. Since the selection of the YAM core is assigned to the first receiver that joins the group, data from every source/receiver, including the shared core, is used to populate the table. In this compilation of runs, the Fan-Out and FCFS metrics had the highest *High Mean Delay* – 14% worse than the highest average produced by the Hops metric. However, the average

of all the simulation runs shows that all the metrics were within 3% of each other. These measurements are within the context of many-to-many communication.

Metric	High/Difference	Average/Difference
BW	0.3132 s (07%)	0.1438 s (02%)
Delay	0.2932 s (01%)	0.1455 s (03%)
Fan-Out	0.3325 s (14%)	0.1406 s (00%)
Hops	0.2916 s (00%)	0.1409 s (00%)
FCFS	0.3325 s (14%)	0.1406 s (00%)

Table 1: YAM - All receivers

Figure 21 below presents a step-by-step display of the average delays as each new receiver joins the group. Just as in Table 1, the delays represent measurements obtained in the case where all receivers are acting as sources to the group. Given the fact that the receivers are randomly chosen from the boundaries of the topology, the average delay increases for the initial set of new receivers due to the fact that these initial branches tend to span the network. However, as the receiver set grows, the length of the new branches decreases because they are grafted onto existing and relatively close branches.

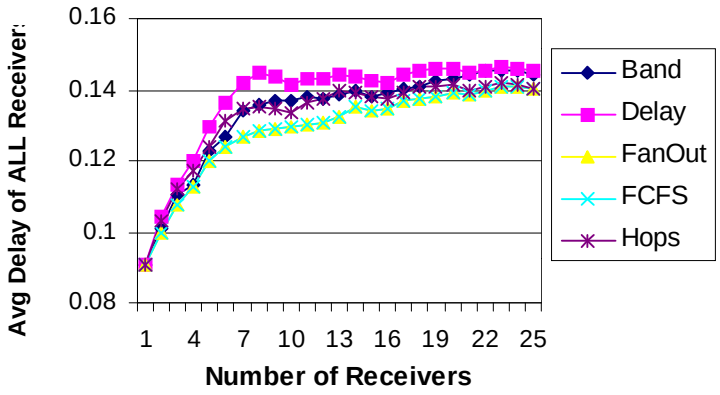


Figure 21: Average Delay as Receivers Join the Group

Table 2 represents CBT. Unlike YAM, CBT arbitrarily selects a node as the shared core, which one can presume is not a group member. Hence, by using YAM to construct the tree, and yet

excluding data sent from the first receiver (i.e., the elected core[19]), it can be argued that the resultant data distribution tree represents one produced by the CBT protocol, since both YAM and CBT produce bi-directional trees. Admittedly, this slightly alters the comparison between a YAM-tree and a CBT-tree since there is one less set of data point to populate the table. But this slight discrepancy is not seen as a dominant factor since the purpose of this exercise is to examine different criteria as opposed to different protocols.

As in the case of Table 1, measurements are presented for all 5 types of criterion, as opposed to just using the Hops metric. This is done to provide a quantifiable measure of how CBT can be improved if it were to use other metrics for building its tree. The values for the *Highest Mean Delay* is the same for Table 1 and 2. However, the subtraction of data originating from the core, increases the *Average Delay* associated with the CBT tree in comparison to the YAM tree. In Section 5.7 below, I show this difference to be significantly greater when investigating the worst case scenario of shared tree protocols that arbitrarily elects the core for the group.

Metric	High/Difference	Average/Difference
BW	0.3132 s (07%)	0.1442 s (02%)
Delay	0.2932 s (00%)	0.1472 s (05%)
Fan-Out	0.3325 s (14%)	0.1408 s (00%)
Hops	0.2916 s (00%)	0.1424 s (01%)
FCFS	0.3325 s (14%)	0.1408 s (00%)

Table 2: CBT - Core is not a Receiver

Table 3 is a very general representation of a tree built by the Express Multicast protocol, wherein only a single node acts as the source of data for the tree. In the simulation, this representation is accomplished by using YAM to construct the tree, but only using results obtained by sending test data from the Core to the receivers.

As expected, the Delay metric produces the lowest value for both *High Mean Delay* and *Average Delay* measurements. Unlike the measurements for YAM and CBT, Table 3 shows a large variance in both *Average* and *Highest Delay* for metrics other than Delay and Hops. This can be attributed in part to the reduction in the number of data points comprising the measurements; i.e.,

[19] Note: the CBT core is located at the edge of the simulated topology.

only measurement between core and receivers. More importantly, the increase in delay in comparing Fan-Out with the Delay metric shows that Fan-Out can be undesirable under certain conditions.

Metric	High/Difference	Average/Difference
BW	0.2350 s (35%)	0.1345 s (31%)
Delay	0.1744 s (00%)	0.1027 s (00%)
Fan Out	0.2542 s (46%)	0.1356 s (32%)
Hops	0.1771 s (04%)	0.1035 s (01%)
FCFS	0.2542 s (46%)	0.1356 s (32%)

Table 3: Static - Core to Receivers

Figure 22 below presents a step-by-step display of the average delay from the core to the receivers as each new receiver joins the group. Just as in Figure 21 above, one sees that the average delay increases for the first few receivers that join the group. However, after the eighth receiver, the delay tends to remain relatively level as the rest of the group members graft branches onto the tree.

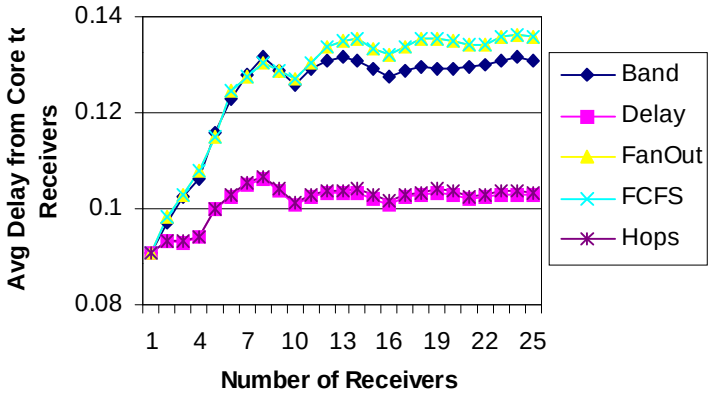


Figure 22: Average Delay, From the Core, as Receivers Join the Group

One particular difference between Figure 21 and 22 is the early and consistent disparity in average delay in Figure 22 between the Hops and Delay metric versus the other metrics. Again, the reason for this disparity, as in the case of Table 3, is that both the Hops and the Delay metric use the core as the reference point in selecting the best path. Given that the propagation delay of the links is relatively narrow, then as expected, the Hops metric produces similar results to that of the Delay metric (which would always be expected to display the best results in this case). Even

though the Bandwidth metric uses the core as a reference point, its range within the simulated topology is somewhat larger (in comparison to the Delay metric) and its selection criteria make it orthogonal to the evaluation criteria (i.e., the lowest average delay from core to receivers). Hence given this orthogonal perspective, the Bandwidth metric, as well as the FCFS and FanOut criteria, are shown to produce worse results than that of the Hops and Delay metrics.

Finally, Table 4, shown in section 5.5.2, represents general information about the control message cost and the cause & effect in how the YAM protocol constructs a tree. The first column, ROUTER-state, indicates how many routers have state for the multicast group. The second column indicates the average number of nodes in the network that receive a 1-to-Many Join for each of the 26 receivers of the group. This information is also presented in Figure 26 so that the reader can view the information on a case-by-case basis as each receiver joins the group. The third column reflects the average number of responses received for each 1-to-Many join that is sent.

5.4.1 Delay from Other Sources

The above set of information focused on one-to-many and many-to-many communication, with the former case based on the core as the reference point for the single source of data. In this subsection, I use three other reference points from which to measure average delay: the 6'th, the 11'th, and the 18'th receiver that join the group. Specifically, I designate these receivers as additional sources to the group and measure the average delay of all receivers as each new receiver joins the group. The purpose of this information is provide an additional perspective of the various metrics, some of which use the core as the reference point in which to graft a new branch. However, it should be noted that the selection, while arbitrary, is still somewhat subject in comparison to the single-source and many-to-many schema presented above.

Figures 23, 24, and 25 show the average delay experienced from receivers/sources 6, 11, and 18, respectively of each simulated run. One will notice that the average delay is either worse or better from a different nodes' perspective – which is to be expected given that YAM does not attempt to produce optimal bi-directional trees for all possible sources.

An interesting aspect about all three figures is that the FanOut and FCFS metrics/criterion consistently produce the lowest average delay in comparison to metrics like Hops and Delay. As discussed in the earlier subsection, this consistently better performance of FCFS/FanOut is attributed to the fact that these metrics do not use a specific point of reference in constructing the tree. Hence, branches are a bit more 'centered' in comparison to the rest of the receiver set, as opposed to having all branches directed to the single shared core (as in the case of the Delay and Hops metrics).

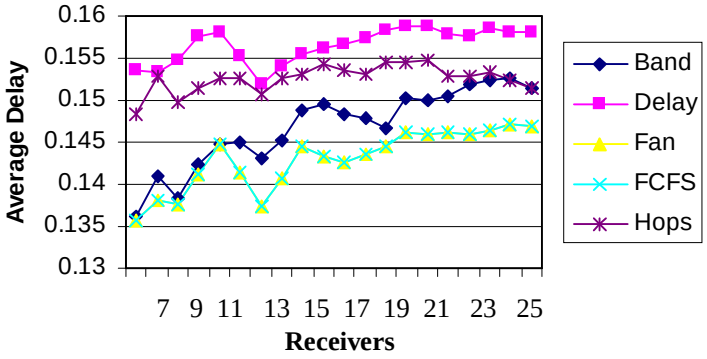


Figure 23: Average Delay From 6'th Receiver

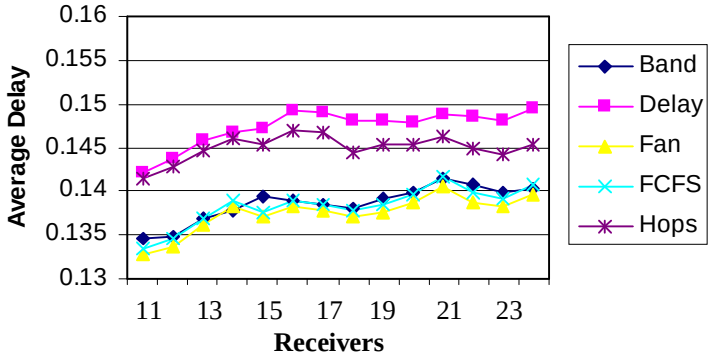


Figure 24: Average Delay From 11'th Receiver

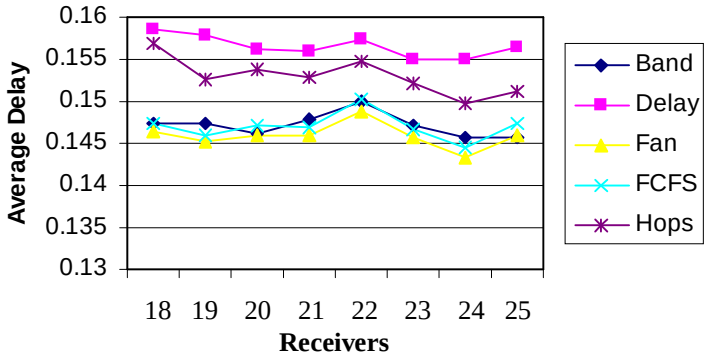


Figure 25: Average Delay From 18'th Receiver

5.5 Observations

For YAM and CBT trees (Table 1 & 2), most of the metrics produced roughly the same average delay for many-to-many communication. This is expected because both protocols produce bi-directional trees. An interesting aspect is that the best average delay (i.e., the lowest), though only a slight improvement, is produced by the Fan-Out and First-Come-First-Served metrics. A reason why the Delay metric is not substantially better than the other metrics is because its focus is on the shared core, which is always placed somewhere at the edge of the topology. Though not shown, preliminary tests have shown that cores selected from a more topological center of the network will produce an average delay for the Delay metric that is either as good or slightly better than the Delay metric.

In the case of the single-sender Express multicast tree, Table 3 shows that the Hops/Delay metrics have a significant improvement in average delay versus the other three metrics. This is understandable since: a) the source and the shared core are the same node, b) the Hops/Delay metrics build minimal cost branches between the receiver and the core, c) the link propagation values of the topology have a small deviation factor, and d) Table 3 only uses one source (the core) to generate results. Even though the Bandwidth metric constructs its branches based upon minimal cost between the receiver and the shared core, the variance of its metric value is orthogonal to that of the Delay metric. Under existing test conditions where load is not a factor in

constructing the tree, and delay is used as a means of examining trees, the Bandwidth metric can only produce a tree as good or worse than the Hops/Delay metrics.

5.5.1 Control Message Cost: Analytical Representation

Representing the cost to the network in terms of the distribution of control message is dependent on the specific design of a protocol. In the case of QoSMIC, [38] divides the discovery mechanism into two processes. One process involves a scoped one-to-many join with a bounded TTL value that reaches only a limited number of nodes. An additional process involves a unicast query that is sent to a proxy node that informs on-tree nodes to issue a unicast join to the leaf router that is attempting to graft a branch onto the shared tree. As such, [QoSMIC] represents the cost of its protocol in terms of:

$$\mathbf{Q}_{cost} = w(w\text{-}1)^{i-1} + T$$

Where **(w)** represents the average degree of a router, **(T)** represents

the number of nodes comprising the multicast tree, and **(i)** represents the TTL value of the one-to-many join. It can be argued that the **(T)** value should be augmented to represent the number of nodes used to forward unicast messages from on-tree nodes to the leaf router (i.e., the number of nodes along a unicast path). However, the approximate complexity of this notation provides us with a foundation from which to represent the cost of other protocols.

In the case of a dense mode protocol like PIM-dm, one can represent the cost <u>per source tree</u> as:

$$\mathbf{PIM}\text{-}dm\__{cost} = w(w\text{-}1)^{m-1} <= (N\text{-}1)$$

Where **(m)** is the maximum TTL value needed to span the minimal path

between the farthest two points or nodes of the network. Given that **(N)** represents the total number of routers in the network, the cost from a PIM-dm protocol never exceeds that of the size of the network. Note: in this case, I do not take into consideration the continual cost of maintaining soft state over time.

In the YAM protocol, a single one-to-many join is sent with an unbounded TTL. This message is terminated by either edge of the network, which has no more downstream routers, or by on-tree

nodes that sent unicast responses to the leaf router that initiated the one-to-many join. As a result, the number of nodes that comprise an existing tree has a corresponding effect on the cost of the tree. I refer to this correlation of on-tree nodes with the nodes of the network in terms its density [106] and represent this as:

$$\text{Density (D)} = \frac{\alpha}{\mathbf{L}}$$ Where (α) represents the total number of links on the tree and **(L)**

represents total number of links in the network. Therefore, as the tree grows in terms of branches and receivers, its density grows in proportion. In taking into consideration the density **(D)** factor, I can now represent the cost of a YAM tree in terms of:

$$\mathbf{YAM}_{\text{cost}} = w(w\text{-}1)^{((1-D)m-1)}$$

Thus, as the density increases due to additional receivers and their corresponding branches, the size of the maximum TTL is correspondingly reduced. This is because each additional branch effectively partitions the area in which a one-to-many join can be distributed into a smaller areas. The following subsection presents simulation results that provide a more quantifiable representation in representing the cost in control message overhead associated with the YAM protocol.

5.5.2 Control Message Cost: Simulation results

Table 4 presents information related to the cost of discovering and establishing a tree for a given metric. On average, and with 46% of the ' edge' nodes of the topology joining the group, between 34.2% and 38.1% of the entire topology receive a one-to-many join message for each new member. A plot for each receiver is shown in Figure 24, wherein the Y-axis represents the average number of nodes receiving a one-to-many join message, and the X-axis represents a time-sequential set of nodes joining the group.

As the number of group members increase, the average number of nodes receiving a one-to-many join message decreases. Variations are attributed to the random location of the receivers joining the group. As mentioned previously, the reason for the decrease is that when an on-tree node

receives a one-to-many join message, it terminates further redistribution and sends a unicast response. It should also be noted that between 5 to 7 nodes/receivers do not send a one-to-many join message. This is because they are already on the tree due to downstream neighbors that have previously joined the group.

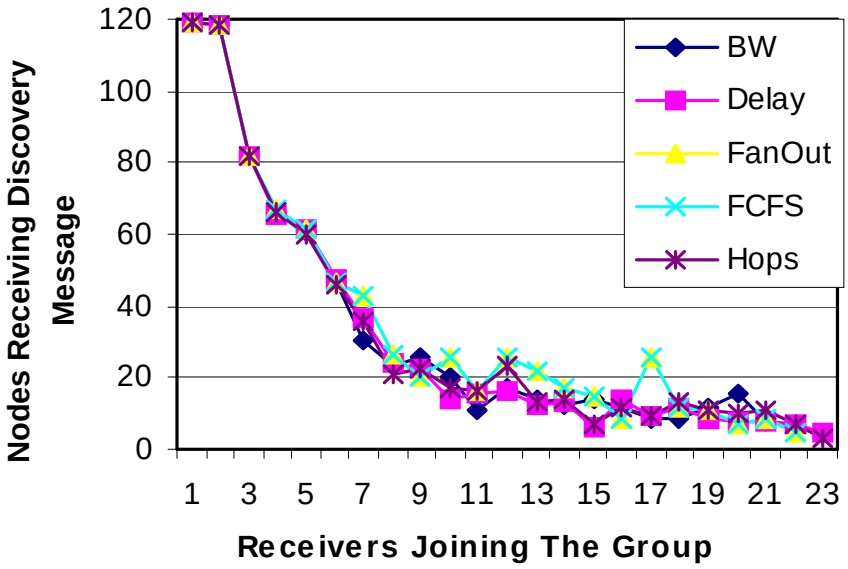


Figure 26: Average Cost

Table 4 also shows that the average number of responses of all 40 runs per one-to-many join ranges between 4.3 and 4.6. In other words, these nodes received roughly 4.45 number of paths or choices from which to select the 'best' for a given metric. Though not shown, this number is also a function of the size of the tree in terms of the number on tree nodes, and the location of the new group member in relation to the location of nearby branches.

Metric	State/Difference	Discover	Response
BW	32.5 (23%)	34.5	4.5
Delay	41.5 (57%)	34.2	4.3
Fan Out	26.4 (00%)	38.1	4.6
Hops	37.4 (42%)	34.5	4.3
FCFS	26.4 (00%)	38.1	4.6

Table 4: Information in Constructing YAM trees

One of the most important costs in the construction of a shared tree is the number of non-receiver nodes (routers) that maintain state for the group. Table 4 shows that the Fan-Out/FCFS metrics have state in an average of 26.4 routers, while the Bandwidth, Hops, and Delay metrics have between 23% to 57% more routers comprising the tree.

5.6 Conclusions

Using 1-to-Many Discovery messages generates a significant cost to the network in terms of control message overhead. And certainly, this cost is higher than traditional shared tree protocols like CBT or PIM-sm. However, as shown above, one can characterize this cost as a function of the density of the group. In simpler terms, one can represent the cost in terms of **O(*n*/*r*),** where ***r*** is the number of receivers and ***n*** is the number of routers in the network. I base this postulation on a couple of factors. The first is that on-tree YAM nodes do not forward 1-to-many discovery messages for the group that it is a member of. This means that as branches are grafted onto the shared tree, they effectively partition the network into smaller sub-regions. Given that in the simulation environment group members are chosen from the edges of the topology, segmentation is effective in reducing cost as group membership increases because some number of branches will span the edges of the network. Though not shown, the correlation of cost to the number of group members also applies when group members are topologically close. The difference is that the rate of decay in cost per receiver for dense receiver locations is less than that of sparse receiver locations.

Regarding the comparison of different metrics, the simulation results show that the number of routers comprising a shared tree is smaller for Fan-Out/FCFS than for the Hops/Bandwidth/Delay metric. The reason is that both Fan-Out and FCFS promote the formation of trees with high fan-out towards the leaves of the tree (a hub-spoke topology). The other metrics focus on path information between the shared root and the receiver, and thus encourage the formation of trees resembling a star topology – i.e., initial high fan-out close to the root of the tree.

An interesting aspect is that for different metrics, there is no significant penalty, in terms of mean average delay, for many-to-many communication. The only scenario in which delay does become a factor is the case of the single source attached to the shared core (Table 3). However, it

can be argued that the disparity in average delay is only relevant to interactive applications. Lecture or single source type applications, like television/radio, are expected to have little to no interest in variances of delay amongst the group of receivers.

From a slightly different though related perspective, monetary costs associated with star topologies can be of high importance because of the economic charging model of trees that span many transit domains. Separate conversations in [17, 73] discuss ideas in how Internet Service Providers (ISPs) can charge for multicast sessions. In [17], the concern related to the fact that inter-ISP multicast sessions place a monetary burden on upstream ISPs because they are charged cumulatively by the downstream ISPs. On the other hand, local sessions involving a single ISP and its directly connected stub clients is less of a concern in terms of billing because the single transit domain may be overprovisioned and not be concerned about recouping the expenditure of its internal resources. [17] discussed a potential integration of charging models that combines the receiver count obtained by protocols like Express for the leaf ISP, and a cumulative charging scheme along the axiomatic model presented in [55].

5.7 Comparing Source Tree and Shared Tree Algorithms

This section takes a slight departure from the work presented in the previous sections of this chapter and makes a brief comparison between shared tree and source tree protocols. The purpose of this comparison is two-fold: a) to show a worst-case scenario with a priori selection of the core, and b) to quantify the difference between YAM, CBT, and DVMRP when the group members are densely populated in one area of the topology.

The simulated topology and connectivity of the network is the same as that shown in Figures 15 through 18. The density in group membership is achieved by randomly choosing receivers from the top four rows of the simulated network. The worst case scenario for a priori core selection protocols like CBT is realized by placing the shared core at the bottom row of the network topology (the last row), thus causing the construction of very sub-optimal trees.

Table 5 presents results of 25 simulated runs of CBT, YAM, and DVMRP. As expected, DVMRP, produced the best overall delay. Correspondingly, CBT produced the worst delays because the core was purposely placed far away from the receiver set. The worst case was CBT

using a hops metric, which produced on average an increase of 481% more delay than that of the DVMRP constructed source-tree.

Protocol/Metric	Average	Difference
CBT: Fan Out	0.2283	395%
CBT: Hops	0.2678	481%
YAM: Fan Out	0.0595	24%
YAM: Hops	0.0619	34%
DVMRP: Hops	0.0461	----

Table 5: Comparison of Protocols

The YAM protocol produces a better *Average Delay* than CBT (or other types of a priori core selection protocols) because its shared core is a product of group membership. Thus, if all the group members are densely populated in one area, then the core will be elected from that area.

Though not shown, the density of the group towards the edge of the topology also causes a large number of nodes in the network (approximately 60) to receive one-to-many join messages. This increase means that there is a greater cost to the network in terms of servicing control messages. Thus a tradeoff is presented is using a YAM tree over a series of source-based tree within a densely populated receiver topology: decrease in delay and stored state versus an increase in control message processing.

5.8 Summary

This paper has presented results in simulating the construction of multicast trees using different criteria. Simulations were generated by the NS simulator, which was augmented to include the YAM protocol. The receivers were randomly chosen at the edges of the topology to ensure that portions of the tree would be sparse. In addition, by placing the receivers at the edges of the graph, I produce an overall structure to the topology that resembles today's Internet, where receivers and sources are located in stub domains at the edges of the network.

Initial results have shown that the Fan-Out/FCFS metrics reduces the amount of state stored in the network in comparison to the shortest-hop metric – the latter typically used with today's shared tree protocols. It has also shown that these metrics do not incur any appreciable penalty in terms

of average delay in the case of many-to-many communication. Finally, the results show that the cost of control message overhead, while significant, is still less than that incurred by source-based tree algorithms and decreases in relation to an increase in group membership.

Chapter 6

Simulating the Impact of *Group Migration* on Shared Trees

Introduction

In the previous chapter, I presented a comparison of trees constructed with different metrics. In this chapter, I take this examination one step further and analyze the impact that group migration has on different types of trees. In order to retain a measure of consistency in my examination, I use the same simulator, simulation topology, and link characteristics (bandwidth, and link propagation delay) as that used in the simulation effort described in Chapter 5. The realization of group migration involves changes in group membership (receivers joining and leaving a group) over time.

There are two scenarios in which group migration is simulated. One is loosely based on an actual multicast session. Specifically, I obtain group migration information from a space shuttle-mission transmitted by NASA over the Mbone. The other scenario represents a hypothetical worst case in terms of movement in which densely populated receivers move in near unison from one topological region to another over time. Related to this latter scenario, I introduce augmentations to the original YAM protocol presented in Chapter 3 and examine the usefulness of migrating the core under certain conditions that do not cause duplicate packets or packet loss.

6.1 Related Work

Within the context of examining shuttle-mission sessions over the Mbone, [2] and [49] presented two approaches using different tools. [49] used Mtrace and RTCPv2 loggers to gather information about one shuttle session. The information that was gathered was primarily focused on loss rates for the group and its set of receivers. [2] took a different approach or perspective in its examination and looked at group behavior of a variety of different sessions, which included a

shuttle mission. This examination was based on the Mlisten tool and was divided into four categories: temporal, spatial, inter-session, and intra-session results.

The examination described in this paper is patterned after [2] and gathers information on join/leave dynamics of group members. I take this information and use it as a basis for constructing and evolving shared trees using different metrics.

6.2 Simulation Approach

The examination of different types of shared trees, coupled with changes in group membership, was accomplished with the VINT/LBL/ISI Network Simulator (NS). Models were developed to represent some of the protocol primitives of YAM so that trees could be constructed and branches could be grafted and pruned as a function of group membership.

The approach used to characterize changes in group membership was based on two perspectives; one involving an actual one-to-many session, and the other on a hypothetical scenario involving distinctive topological migration of the group. The purpose of this dual-approach was to conduct the examination of group membership in terms of an actual case-study, as well as a target of a worst-case scenario. For the real-life session, I chose a recent transmission of a NASA shuttle mission on the Mbone. For the other spectrum, I designed a scenario that represented distinctive topological movement of group members.

The qualitative measure of the performance of the tree, and its evolutionary structure, was based on the end-to-end mean delay of data sent from a source to all the receivers. These measurements are presented from two perspectives. The first involves the delay from data sent by the core to all the other receivers. The other involves gathering the average delay of all sources to all receivers (i.e., many-to-many communication).

- **Case Study: Space Shuttle**

In choosing an Mbone space shuttle session, my purpose was to obtain a framework that presents an approximation of group membership. Specifically, group membership was determined as a statistical approximation based on the reception of membership reports received over 15 minute periods. If a group member sent an RTCP report during this time period, it was considered to be

a member of the group. If no reports were received, then it was assumed that the receiver had left the group -- either by choice, or due to a partition of the Mbone.

One aspect that was not derived from the logs obtained in the shuttle session was the topological location of group members. The selection of group members in my simulations were randomly chosen from the edges of the topology. This provided us with a high probability that certain branches would be of a minimal depth of 8 hops -- representing a measure of sparseness in the location of group membership.

- **Worst-Case Scenario: Hypothetical**

This case presents a hypothetical scenario in which most of the simulated receivers are topologically close to each other and collectively move to other areas. In this manner, I present a worst case scenario for shared trees in that collective movement of the group can adversely affect end-to-end delay of data due to the sub-optimal location of the shared core. So even if the shared core is placed in an initially ideal location that is in the same location as the densely populated receivers, the movement of the group membership will result in a core that is far away from the receiver set. This poor location of a core can adversely affect the performance of a bi-directional tree in terms of end-to-end delay, and is particularly acute for uni-directional trees [9].

- **Simulation Environment**

The topology of the NS simulation was comprised of 120 nodes whose bandwidth and delay characteristics were randomly set between 1Mb/s and 3Mb/s, and 0.05 seconds to 0.14 seconds, respectively. In the Shuttle-case, the initial group membership list was comprised of 30 nodes. At each period, a set of nodes were removed from the list, and then a set was added to the list. The set that joined and left the group at each period corresponded proportionally to the set that joined/left an actual shuttle session transmitted over the Mbone.

In the hypothetical-case, the initial group was set to eight nodes. After each period, seven of the nodes left the group, and seven new nodes were added to the group. The seven new nodes were selected from a different topological region used to select the previous set. In each case, the same set of nodes were used for each type of tree (e.g., delay, bandwidth) that was constructed by the simulator. In addition, each set of nodes joined/left the group at the same time for each criteria.

Thus the only difference between each simulated run was the type of criteria or metric used to construct the tree.

6.3 Baseline Results

The primary means used to measure the quality of a distribution tree involved measuring the mean end-to-end delay of a datagram sent from the source to the receivers of a group. This measurement was taken at the end of each period, when members had finished joining/leaving a group. Two perspectives were used to measure this delay. One involved all group members sending data to all other receivers: the *Many-to-Many* case. This paradigm is used by Distributed Interactive Simulation (DIS) applications and reflects a stressed scenario for multicast routing due to signaling and data emanating from all receivers of the group.

The other perspective in measuring latency involved a single node sending data during the lifetime of the session[20]: the *Single-Source* case. This model represents lecture (or television/radio) type applications that never have other group members send data to the group. In both this and the DIS model, the latency is expressed in terms of the mean value of delay from all receivers for the source(s). Admittedly, these two models are opposite sides of the spectrum in relation to the types of applications that would use multicast technology to distribute data. However, attempting to exercise a model that represents a compromise between the two can be problematic without closely following all the parameters of a real-life session concerning its group membership as well as its exact topology. For now, I leave this as an exercise for future research.

Beyond using latency as a measure of tree quality, I also measure the size of the tree in terms of the number of on-tree nodes that have state for the group. The previous chapter showed that there can exist cases where metrics like Fan-Out can produce a smaller tree (one that has less routers that comprise it) than the shortest-hop or shortest-delay metrics

The rest of this section presents graphs and tables that discuss specific results in producing a tree based on similar membership characteristics as that observed in the Mbone shuttle session. Given

[20] For the sake of argument, we discount the generation of RTCP control messages; which amounts to many-to-many communication.

that my simulated topology has only a 120 nodes, I have reduced the initial group size from the 78 members of the shuttle session to 30 members for the simulated session. In addition, the number of receivers that join and leave a group at each period, reflects a corresponding percentage observed for the shuttle session, as opposed to the shuttle's actual number members that of joined or left.

Finally, the simulation is run for twelve periods – each period corresponding to a 15 minute time span representing a 3 hour session. This is in contrast to the shuttle mission, which was observed for an 8 hour period. The reason for this shorter time period in simulation was because the last 5 hours of the shuttle mission where nearly identical in terms of the final number of members having joined and left the group -- thus not significantly contributing to the final simulation results.

Note: in all of the simulations, the Fan-Out and First-Come-First-Served (FCFS) metrics produced the same topological trees. As such, the following results only explicitly mention Fan-Out, although the reader should assume that the I refer to both Fan-Out and FCFS.

6.3.1 Shuttle-type Simulation

Figure 27 represents the actual join/leave membership characteristics of a shuttle session transmitted over the Mbone. It presents a graph that shows for each 15-minute period: a) the number of group members, b) the number of receivers that have joined the group, and c) the number of receivers that have left. Over this 8 hour period, there was a total of 190 receivers that at one time joined the session. For a specific period, there was a maximum number of 78 group members (in Period-1) and a minimum of 26 members (in Period-24). Nearly 20% of members in Period-1 sent at least 1 group membership report for each subsequent period. An additional 15% went through the process of joining, leaving, and then rejoining the group. This can be attributed to either ' surfing' or temporary breaks in Mbone connectivity that lasted beyond a 15 minute period. The final approximately 65% of the members were only members of the group for a single and distinct portion of the 8 hour observation.

The 190 receivers came from 92 distinct stub domains. Some of the geographical location of these stubs included the U.S., Malaysia, Germany, Spain, U.K., Finland, and Japan. Of the 92

stub domains, there were 42 distinct leaf ISPs identified. Hence, two levels of aggregation existed for the session: approximately 2 receivers per stub domain, and 2 stub domains per leaf ISP. This topic aggregation is discussed further in section 6.4 below, and in Chapter 7.

The information on group membership was gathered between 16:00 and 24:00 Eastern Standard Time (EST) on November 5' th, 1998. From this information I then produce a corresponding proportional set of group members for the NS simulation; using 30 as the maximum group size.

The group members are randomly chosen from the edges of the topology. Figure 27 presents a graph representing some of these characteristics. It shows the current number of group members, as well as the number of members that have joined and left a group for each period. As mentioned previously, the simulation in changes of group membership correspond to just the first 12 periods shown in Figure 27.

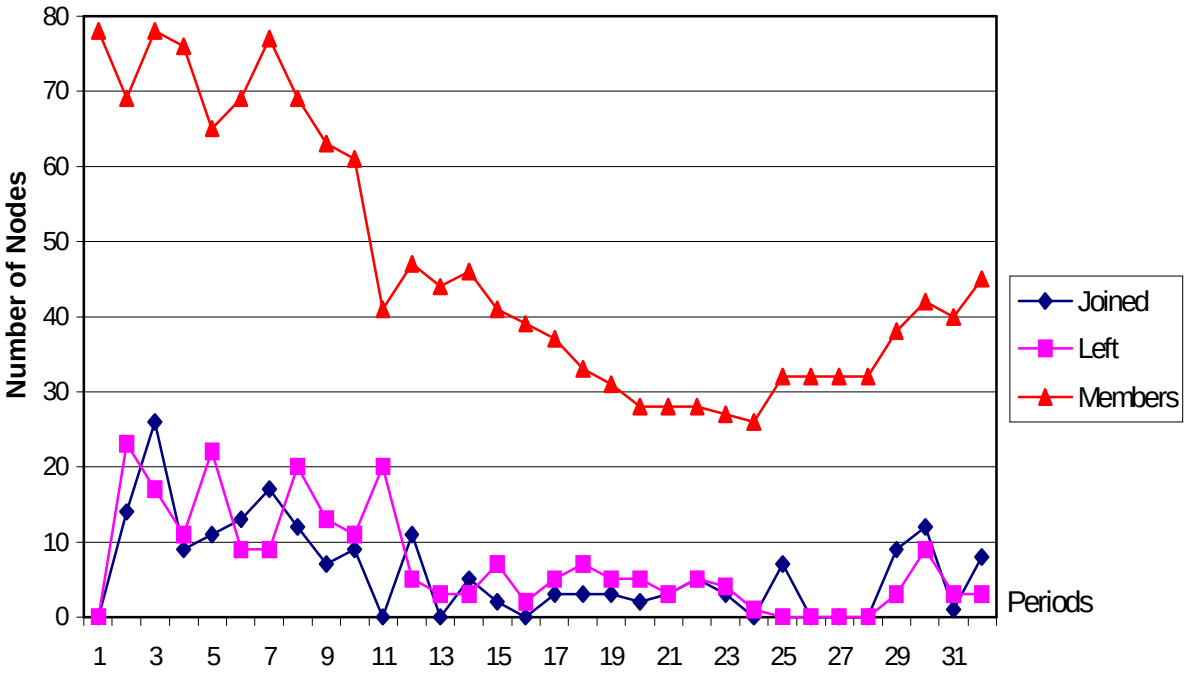


Figure 27: Data Extrapolated from Shuttle-Mission Session

Figure 28 represents results from the *Many-to-Many* case, in which the average end-to-end delay measurements are taken for each period and whereby all members send traffic to the group. As

shown in the figure, the Fan-Out metric generally produces the lowest average delay for each period. This is due to the fact that it tends to graft a branch onto the nearest on-tree node without respect to the location of the core. And while there were some differences amongst the metrics, the differences are not significant in the form of major spikes in delay at any given period.

The average difference of the best performing metric at any given period versus the other metrics was only a 9.4% increase in delay. The worst mean delay occurred in Period 11, where the Delay metric produced 20.5% more delay than the Bandwidth metric. While, this may seem odd at first, given that these metrics use the location of the core as its focus in constructing a tree, it should be noted both metrics are completely unrelated in the criteria used to construct a tree. Hence, while one link may have a low propagation delay, it may also have low available bandwidth, thus making it desirable for one tree/metric and undesirable for another.

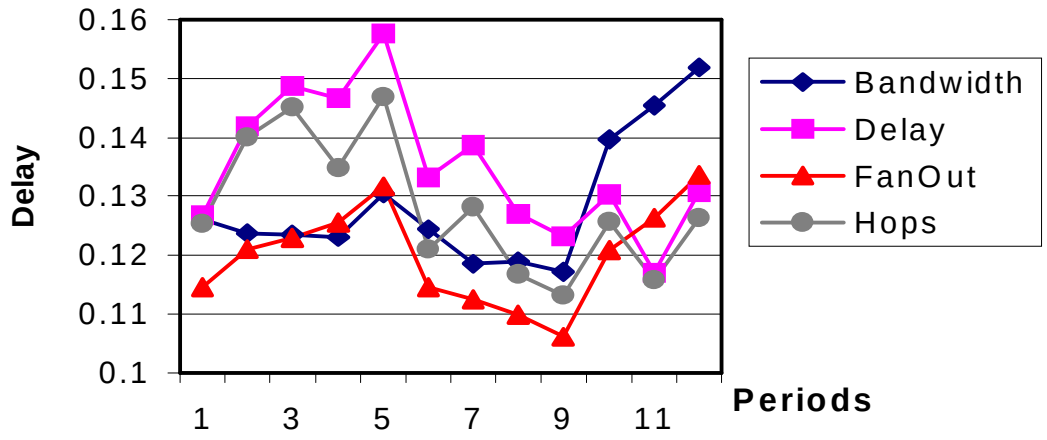


Figure 28: Average Delay of *Many-to-Many* Case

In the example of the *Single-Sender* case for the same simulation run, again, there is no major difference in average delay between the different criteria: the mean difference was 11% between any metric for any given period. Also, since the measurements in this case are taken from only the core to the receivers, then as expected, the delay metric displays the best performance since it is based on having receivers choose the path with the lowest delay between itself and the core.

Figure 29 shows a graph that plots the delay of each metric for each period. One interesting aspect to observe is the near similar pattern that each metric has in relation to each other. With respect to the Delay and Hops metric, each displays near equal plots of delay. This is due to the fact that range in propagation delay is somewhat small, thus variations in hops has minimal effect the end-to-end delay between the core and the leaf router.

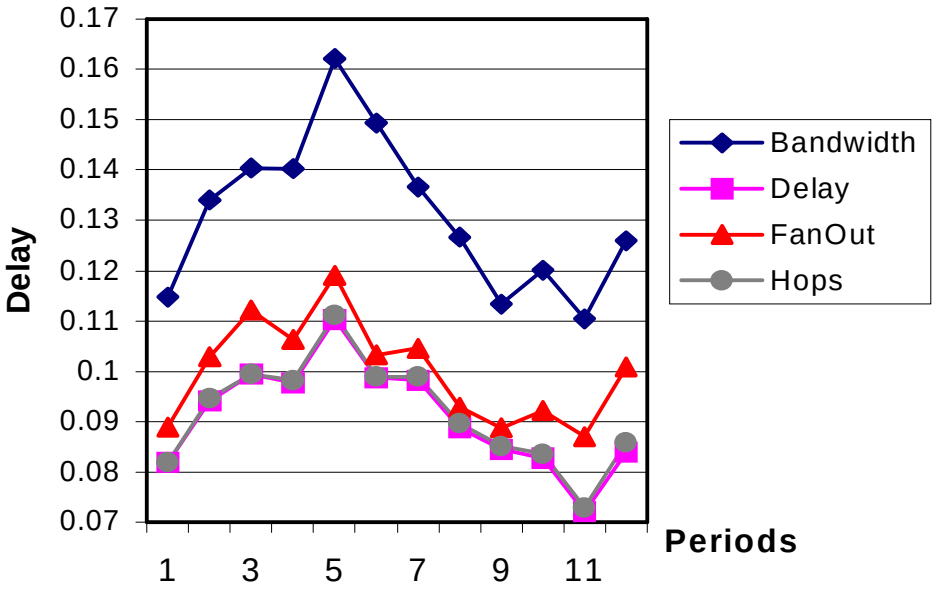


Figure 29: Average Delay of *Single-Source* Case

Having shown the changes in delay in conjunction with changes in group membership, I now present measurements of router state. Specifically, I mean the number of nodes used to form the tree for each metric at each time period. Table 6 shows the number of nodes used to comprise a tree for each period[21] for the shuttle-type simulation. In this case, the Fan-Out metric consistently produced the best savings of state -- i.e., the least number of routers were needed to construct the tree. The average increase in the amount of routers maintaining state for other metrics in comparison to the Fan-Out metric was 25.4%. The worst case occurred at Period 7, in which the Delay metric produced a tree that had 48.3% more routers than Fan-Out.

[21] the number excludes existing group members that may be upstream from other receivers.

Period	Bandwidth	Delay	*FanOut*	Hops
1	26	34	**20**	33
2	24	40	**22**	35
3	24	36	**21**	32
4	24	42	**23**	32
5	27	44	**23**	31
6	24	28	**18**	23
7	23	37	**19**	28
8	24	32	**20**	25
9	24	32	**20**	25
10	30	36	**23**	28
11	27	30	**22**	25
12	26	34	**21**	25

Table 6: Number of Routers per Tree-Type

On a related note, the Hops metric produced trees that had on average 27.3% more routers than those produced by Fan-Out. This is particularly of interest given that today' s multicast routing protocols are pretty much based on shortest hop metrics. Figure 30 below plots the values of table 5 onto a graph to display each metric in relation to each other over the 12 periods. An interesting visual characteristic of the Fan-Out metric is that it does not oscillate with any appreciate difference over the 12 periods. The Delay metric, on the other hand, does display some measure of change as the group membership changes over time.

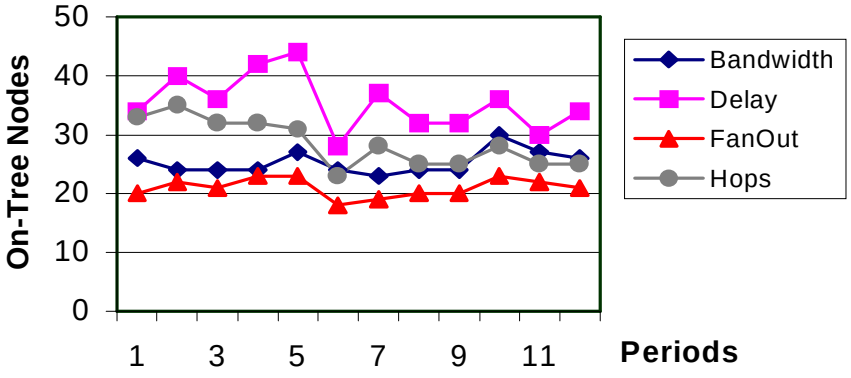


Figure 30: Number of On-Tree nodes per Simulated Period

6.3.1.1 Observations and Extrapolations

In Figure 26 from the previous chapter, I showed a graph representing the cost, in terms of control message overhead, to the system as receivers join a group. This cost starts off initially high for the initial two members of the group – approximately, **O(N)**, where **N** represents the number of network nodes in the graph. As each subsequent receiver joins the group, the cost is reduced such that it resembles **O(N/R)**, where **R** represents the number of receivers. Hence, we have a baseline representing the worst case scenario whose impact decreases as group membership increases[22].

The examination of the Space Shuttle session and its group membership characteristics allows one to make extrapolations in the context of deploying YAM into an internet. Recall that during the 8 hour period of monitoring the session, 190 receivers joined the group. These receivers came from 92 distinct stub domains, which in turn were provided Internet connectivity by 42 leaf ISPs. This means that each ISP represented an aggregation of 4 receivers. Hence, if one were to correlate receivers to signaling, then instead of 190 distinct 1-to-Many joins for the group, one could reduce this to 42 distinct 1-to-Many joins for the shuttle session. This abstraction is based on a design paradigm of YAM that states that regions serviced by a Joining Node (the entity that issues the 1-to-Many join) are abstract and can correspond to routing areas that are larger or smaller than stub domain. If one chooses the former case, then the region can be comprised of a leaf ISP and its accompanying stub domains, thereby presenting a broad level of aggregation in signaling beyond that of simply the accumulated receivers of a domain.

In the context of Internet as it is today (roughly, summer of 1999), there exists 7,300 ISPs in the world today that services over (a conservative estimate of) 2 million domains[23]. Placing JNs at leaf ISPs would in turn constrain the initial impact of the 1-to-Many join to just the set of 7,300 domains, as opposed to the million of stub domains in today's Internet. And, of course, this impact is further reduced as group membership increases across the Internet, which would be expected given a group that has been administratively scoped across the entire Internet. In cases where the group is administratively scoped for a smaller area, such as the European continent, the 1-to-Many join would be correspondingly constrained.

[22] This does not take into account design schemas meant to help reduce the impact of the 1-to-Many Join.

[23] Current information on ISPs can be found at http://thelist.internet.com, while information on domains can be found at http://www.isc.org

6.3.2 Hypothetical Case

In a separate set of simulation experiments, I examine a scenario that involves the topological migration of group members through a network. In each period, a group of receivers that are topologically close to each other joins the group. At each subsequent period, most of the current members leave the group, and a new set of receivers located at a different area joins the group. This migration scenario is exercised under two different conditions: one in which the elected core is static and never moves through the life of the simulation, and the other is where the core migrates in relation to group membership and under certain conditions. The former case is a reflection of existing shared tree protocols, wherein once the core is selected, it does not change as a function of the dynamics of group membership. Arguments as to why this is the standard case center on the fact that data can be lost or duplicated while the core migrates and downstream nodes reconfigure themselves to prune the old branch and graft a new one to the new core.

As stated previously, YAM elects the core based on existing group membership. The easiest rule to accomplish this is to have the first group member be elected as the core. A rationale behind this is that the first receiver acts as a sink for all data sent to the group – i.e., the initial receiver as well as the initial branch has a measure of value in terms of the data sent through it. This measure of value is diminished to some degree if the initial receiver leaves the group. The value is diminished considerably further if there exist only one downstream link from the core to other group members – i.e., there are no other branches emanating from the initial core. This is because data is sent to a node simply because it is the core and yet is dropped by the core because it is not a receiver and there are no other downstream branches to receive it.

To address this problem, the YAM models use two new Information messages to elect a downstream on-tree node as the new core. The first message, Elect-Core, is sent down the tree by the core when it is no longer a receiver and when it has only one downstream interface. If the downstream node is also not a receiver and has only one downstream interface, it forwards the message to its single child. Note that each node that forwards the Elect-Core message removes state for that group.

When the Elect-Core message is received by a downstream node that is either a receiver or an on-tree node that has more than one downstream interface, it terminates the Elect-Core message and sends a New-Core message to all the routers that comprise the tree. This new message informs all on-tree routers of the identity of the new core. Note that this series of messages does not cause traffic to existing receivers to be lost or duplicated.

In figures 31-a and 31-b, I present the end result of two trees whose majority of members have moved from Area-1 to Area-4 of the simulated topology. Each tree is constructed using the Fan-Out metric. At Period-1, eight receivers are randomly selected from nodes 0 through 29, located in Area-1. The initial receiver, Node-2, is selected as the core for each tree[24]. In Period 2, seven of the eight group members leave, and a new seven are then chosen from Area-2 – leaving one group member active in Area-1. Later, in Period 3, seven of the group members leave and seven more are chosen from Area-3. Finally, in Period 4, a final seven receivers leave (leaving node-64, the eight node, active in Area-3), and a final seven are chosen from Area-4. In this fashion, there is always one receiver that is active as the location of the group membership gradually migrates from one area to the next.

In the case of figure 31-a, the core is static and never changes location during the lifetime of the simulated session. This example is an inherent feature of protocols like CBT, PIM-sparse, and BGMP and represents the cause and effect of not updating core placement as a function of change in group membership. So while the initial core placement may have been ideal, the final structure of the tree makes the core location appear sub-optimal.

Figure 31-b shows the case in which the core does migrate as a function of group membership. The rules used to accomplish the migration are based on: a) the core no longer being a receiver, and b) the core only having one downstream interface active for the group. This allows the migration to occur with out loss of data packets as well as no appreciable control message or CPU overhead in attempting to elect an ideal core.

[24] The nodes are numbered from 0 to 129. Node-0 is at the top right corner and Node-9 is at the top left corner of figures 29-a and 29-b.

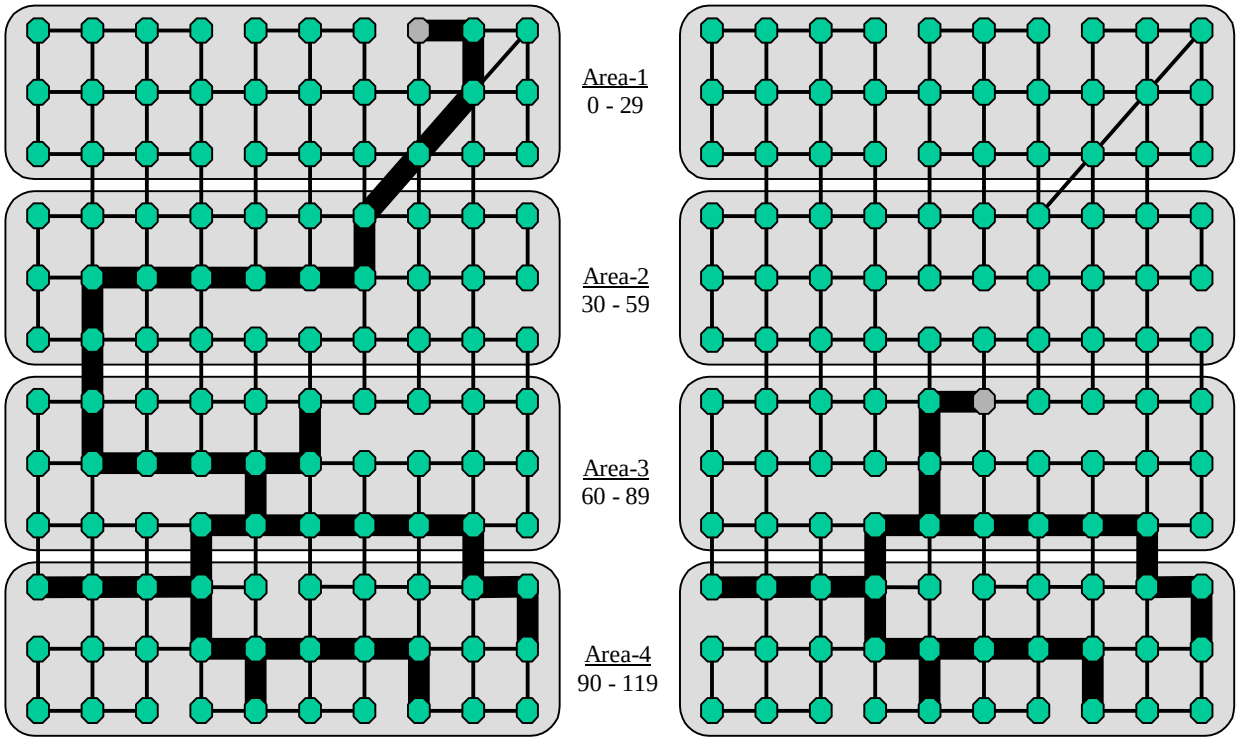


Figure 31-a: Static Core **Figure 31-b: Core Migration**

Figure 32 presents a quantifiable measurement of the two trees shown above. The difference is shown in terms of the average delay for each period after group members have joined the group. The measurements are patterned after the *Many-to-Many* communication case and reflect the average delay of each group member sending traffic to the group. As one can see in this figure, the rate of average delay increases for each period.

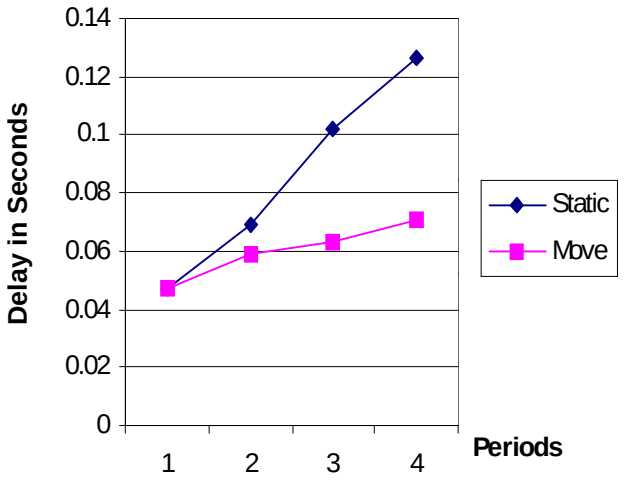


Figure 32: Average Delay of *Many-to-Many* Case for Fan-Out Metric

The increase is due in part to the fact that one group member always exists in the previous region. The difference in the rate of increase between the two sets of plotted points corresponds to the Static-case, in which the core never moves, and the Move-case, where the core is allowed to migrate under certain conditions.

Figures 33 through 35 present a similar case-by-case comparison of the other metrics in terms of the core being either static or moving. One will notice that the disparity between the sets of plotted values for each metric is not as large as that shown in Figure 32, the Fan-Out metric. This can be attributed to the fact that the other metrics (Bandwidth, Delay, Hops) have a common element of constructing their trees in relation to the location of the core. Therefore, since the core exists in a previous region then each leaf router will attempt to graft its branch towards that region. The difference in delay is influenced by how far away the core is to the current set of receivers.

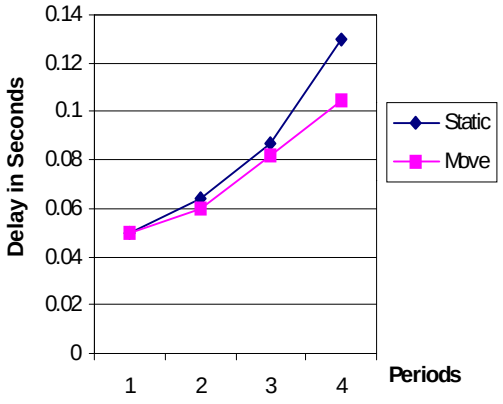


Figure 33: Bandwidth

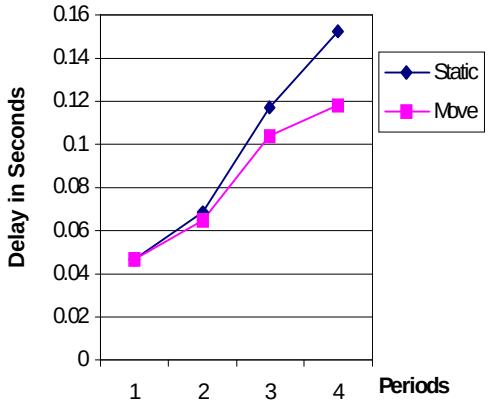


Figure 34: Delay

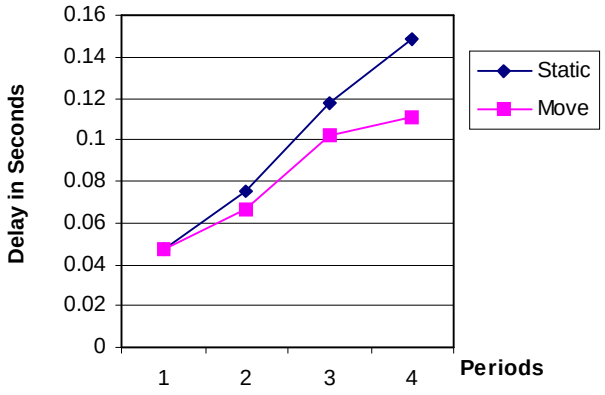


Figure 35: Hops

In the simulations, I have also used state as a means of measuring the quality of a tree. The following figures and accompanying tables represent the amount of state stored in cases where the core moves and when it is static. Table 7 shows an average all the plotted points representing the number of routers for each tree constructed by a different metric. For each metric in Table 7, there is between a 3 to 4 fold increase in state from the first period to the fourth period. Table 8, on the other hand, shows that roughly twice as much state between the initial and last periods.

Since the Fan-Out and the FCFS metrics produce the same result, I only display the output of only the Fan-Out metric.

As in the previous cases, one sees that the Fan-Out metric consistently produces trees that have nearly the same number or significantly less number of on-tree nodes in comparison to other metrics. This applies to either case where the core moves, or the core stays static over a simulated session.

Period	**Bandwidth**	**Delay**	*FanOut*	**Hops**
1	6.7	6.7	**6.0**	6.0
2	12.0	13.4	**9.7**	11.7
3	17.1	25.5	**16.6**	19.2
4	24.1	29.7	**23.5**	23.7

Table 7: Router Count with Static Core

Period	**Bandwidth**	**Delay**	*FanOut*	**Hops**
1	6.7	6.7	**6.0**	6.0
2	9.5	12.5	**8.7**	11.5
3	15.2	18.7	**11.2**	14.9
4	13.5	18.2	**10.2**	14.2

Table 8: Router Count with Moving Core

Figures 36 and 37 provide a visual comparison between the plotted points tables 7 and 8 relating to the average number of routers used to a construct a tree using the Hops and Fan-Out metrics. In both cases, one sees a dramatic change in values starting at period 3, when the set of receivers are selected from the 3'rd region of the topology. In the case of the static core simulation, the core remains in the third region. In the case of the migratory core, it remains in the previous region from where the current set of receivers are selected.

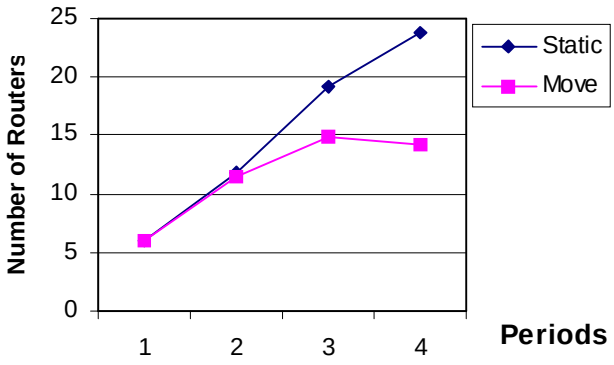


Figure 36: Hops Metric

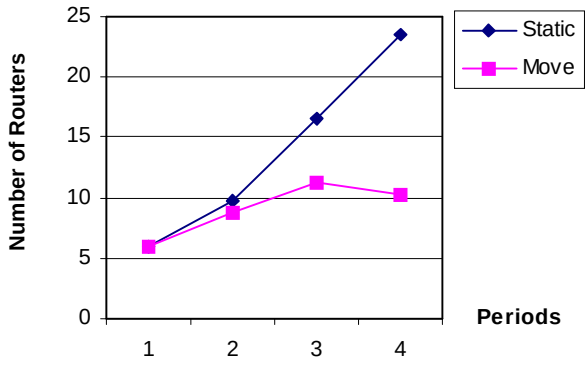


Figure 37: Fan-Out metric

6.4 Summary

I have shown simulation results that examine the impact on shared trees caused by changes in group membership. The examination focused on trees constructed and maintained by one of the following metrics or criterion: a) Delay, b) Bandwidth, c) Fan-Out, d) First-Come-First-Served, and c) Hops – with the last metric being the one used by today's multicast protocols.

The examination was based on the average delay of traffic sent to the group on both a many-to-many basis as well as just the single-sender example (in my case, the core to the receivers). While the simulation results showed that certain metrics outperformed others as group members changed, the difference was not very large at any given period. I also noticed that the Fan-Out/FCFS metric consistently produced trees that were comprised of the least number of nodes – resulting in less state stored by nodes of the network.

Finally, I presented a schema in which core migration can be accomplished with minimal impact to the system. I showed how changing the location of core as group members move from one location to the next can help reduce the increase in mean delay while no data packets destined for receivers are lost or duplicated.

Chapter 7

Conclusion

In conclusion, I reiterate the contributions presented in the thesis and discuss areas of future research concerning my approach to constructing shared trees based on a particular Quality of Service.

The specific contributions of my thesis are:

- I have designed a QoS sensitive shared tree protocol that is capable of constructing trees based on one of several types of metrics. This construction involves a variance of the Greedy algorithm that uses a one-to-many joining mechanism to discover a set of paths from a leaf router to different on-tree nodes. The path that best satisfies the target metric or criteria is the one chosen as the new branch. And perhaps most importantly, I have accomplished this using an underlying unicast routing protocol that constructs routes with a single metric.
- I have also made the structure of the tree, and specifically its shared root, more closely associated with the topological location of its receiver set. This involves making the selection of the core based on the first receiver that joins the group. The approach is in contrast to many of today's shared tree protocol designs that use an a priori election of cores irrespective of group membership. In addition, I add simple rules to my design that allows the core to migrate under certain conditions without loss or duplication of packets.
- I have presented a series of simulation exercises that have provided a quantifiable examination of my shared tree design. I showed how a tree constructed with a Fan-Out metric does as well or better than other metrics regarding average end-to-end delay from data sent by all members of the group. In addition, I have shown how the Fan-Out metric consistently produces 'smaller' trees in terms of the least number of nodes needed to comprise it – thus, extending the measure by which multicast state is reduced in the network. My simulation studies have also shown that the cost with respect to overhead bandwidth is reduced as the number of receivers, or group density, increases.

- Finally, I have produced an implementation of a QoS sensitive protocol that involved alterations of the Unix kernel needed to support native <*,g> state (i.e., multicast forwarding tables that do not rely on an additional entries identifying the source or core of the group). My purpose for this implementation was not to examine the protocol over actual networks, but to identify issues that exist beyond just a theoretical design of a protocol or algorithm.

Part of the motivation of this work was prompted by a need to add a measure of QoS in the construction of a tree in such a way that the problem space is not viewed as a Non-deterministic Polynomial (NP) Complete problem. Previous related work in multicast routing (e.g., PIM) was always constrained by the requirement of constructing trees based with a single metric. In the case of shared tree construction, this constraint was compounded by having the shared core act as the single point of reference in tree construction. However, by having leaf routers view any on-tree node as a reference point in which to graft a branch, I have presented a design that provides an on-demand choice of paths from which a new branch can be added to a tree.

Another motivation that influenced my work involved the exploration of different types of metrics or criterion that one can use to build a tree. Traditionally, unicast routing protocols have constructed routes based on the shortest hop count metric between two edges in a graph. Variations in this theme have dealt with minimal delay or maximum available bandwidth metrics – all of which has been subsumed in existing multicast routing protocols. My thesis has gone beyond this and identified metrics that are attributed solely to multicast routing. In particular, I have identified the Fan-Out metric and used it to promote the construction of trees that have a high fan-out towards the edges of the tree. I have shown the value of this type of metric as one that produces trees comprised of the least number of nodes (in comparison to other metrics). I can also define variations of this theme and attempt to construct trees that have maximum fan-in in cases where there exist many sources and relatively few number of receivers.

- **Future Work**

There are several areas of future work that can spring forth from the baseline contributions presented in my thesis. These involve expanding the capabilities of the YAM multicast protocol,

realizing its potential in relation to new QoS applications, and comparing the design principles of YAM with other work that can be expected to come into fruition.

Having presented the groundwork for constructing QoS sensitive trees, I can now look at how I can add to my work. One area for future research involves the construction of multi-metric trees, in which individual branches are based on one of several different metrics, opposed to one uniform metric for the entire tree. In this case, some receiver sets may attempt to graft branches based on shortest delay to the core, while others may construct branches based on highest available bandwidth. In my scenarios, the average number of responses generated by a one-to-many join was between 4 and 5. Given that the selection was based on an entire path, as opposed to the individual collection of links in a network, the amount of computational time is dramatically reduced. Hence, it stands to reason that using a combination of criteria to select one of several paths should not produce an NP Hard problem to solve.

In the area of core migration, it would be of interest to find out the quantifiable benefits and penalties of taking a more active approach in migrating the core based on changes in group membership. The approach shown in this paper relies on generating the least amount of disruption in terms of packet loss and/or duplication. Given that changing the location of the shared core provides some overall benefit, it would be interesting to determine the tradeoffs of a more aggressive approach in migrating the shared core versus the consequences of packet loss or duplication. There has been previous discussions and simulations about this topic, but there has not been any quantifiable results regarding the adverse impact from either simulation or implementation of protocols designs to support core migration.

Adaptive QoS multicast applications, and their interaction with a QoS sensitive multicast protocol, is a rich topic that would be interesting to investigate. With the standardization of the RSVP protocol, and even its predecessor, the Streams-II experimental protocol [96], has seen the emergence of applications that reserve resources for end-to-end traffic in order to achieve better than Best Effort service. These applications can make specific reservations based on delay or bandwidth or some combination of both. Multicast applications can also be adaptive in their operation and realized by multi-layered distribution of traffic. An example would be the correlation between hierarchically encoded video data that is sent to multiple multicast groups – the culmination of groups producing a higher granularity of traffic.

However, in both instances there is no direct communication with the network in general, and routing specifically, regarding the discovery and instantiation of a branch that better satisfies the desired type and quality of service. One reason for this is that within the context of IP there as been an historic separation of routing information from the end host, or receiver. This is further enforced by the use of IGMP as a separate signaling protocol for determining group membership in the LAN or subnet and thus acting as an added level of separation in multicast signaling.

To go beyond this and achieve a more cooperative integration of QoS applications and multicast routing, one could augment IGMP, and the associated application program interface (API), such that it is capable of conveying requests for different desired QoS to the tree construction protocol. Conversely, it would probably be advantageous to have the network or tree convey unsolicited characteristics of itself (e.g., changes in monetary cost of a branch or core migration). How this would be accomplished, and the granularity of the exchange of information through IGMP, is an open issue. However, one can postulate that it may be advantageous to use a signaling schema that conveyed 'hints' like 'less delay' or 'more bandwidth', as opposed to more granular bounds. I state this because of the perception that a specific value may incur an explosion of state in the network.

Another approach in conveying QoS information is to develop a new LAN-based protocol designed to support host-network QoS signaling communication. Currently, RSVP is the current standard used to support reservation primitives over an IP network. Unlike the multicast model in which a separation of signaling exists between host and network, and amongst routers of the network, RSVP is based on a single end-to-end signaling protocol[25]. By developing a new generic host-to-1'st-hop-router signaling protocol, one could integrate the objectives of QoS multicast, QoS unicast, as well as that of IGMP in order to have a single protocol support a variety of basic functionality.

Concerning the subject of pricing and the construction of multicast trees, I briefly presented the case in which mechanisms like shadow prices, with their feedback schema, can be integrated with YAM in order to refine the selection process of choosing a path as a new branch on the shared tree. However, a more detailed examination is needed to determine the extent and frequency by which feedback is propagated to other nodes of the tree. Given that the metrics defined in

Chapter 3 are unrelated to each other, this examination would need to be done on a per-metric basis. In addition, it would be beneficial to examine how a more economic or pricing based exchange of information can influence the construction of tree for existing and potential group members.

Finally, another aspect that needs to be investigated is a comparison between YAM and a QoS unicast routing protocol. This comparison would center on the cost of the on-demand discovery scheme of YAM versus the periodic advertisements and computation of multiple routes per destination. An initial assumption is that if the number of metrics (producing a correspondingly high level of distribution of control messages), then the cost of YAM and its on-demand discovery of alternate paths may be lower than the cost associated with protocols like QMOSPF.

[25] Recent work, such as [111], has involved an aggregation of RSVP signaling. However, this work focuses on router-to-router communication as opposed to the just host and the network, as in the case of IGMP.

References

[1] R. Ahula, T. Magnanti, J. Orlin, *Network Flows*, pp. 788-801, Prentice-Hall, 1993

[2] K. Almeroth, M. Ammar, *Multicast Group Behavior in the Internet's Multicast Backbone*, IEEE Communications, June 1997.

[3] G. Armitage, *Support for Multicast over UNI 3.0/3.1 based ATM Networks*, RFC-2022, November, 1996.

[4] A. Ballardie, Ph.D. Thesis, University College London, May, 1995.

[5] A. Ballardie, Core Based Trees (CBT) An Architecture for Scalable Inter-Domain Multicast Routing, ACM Sigcomm'93 Conference Proceedings, October 1993.

[6] A. Ballardie, *Core Based Trees (CBT Version 2) Multicast Routing*, RFC-2189, Sept 1997.

[7] A. Ballardie, *Specification for Core Based Trees: Version 3*, Internet-Draft - Work In Progress, October 1997.

[8] D. Bertsekas, R. Gallager, *Data Networks*, pp. 374-376, Prentice-Hall, Englewood Cliffs, New Jersey, 1992.

[9] T. Billhartz, J. Cain, et.al., *Performance and Resource Cost Comparisons for the CBT and PIM Multicast Routing Protocols*, IEEE JSAC, April 1997.

[10] J. Bolot, I. Wakeman, T. Turletti, *Scalable Feedback Control for Multicast Video Distribution in the Internet*, Proceedings of ACM SIGCOMM'94, September, 1994.

[11] R. Braden, D. Clark, S. Shenker, *Integrated Services in the Internet Architecture: An Overview*, RFC-1633, June 1994.

[12] K. Carlberg, J. Crowcroft, *Building Shared Trees Using a One-to-Many Joining Mechanism*, ACM Computer Communication Review, January 1997.

[13] K. Carlberg, J. Crowcroft, *Quality of Multicast Service for Yet Another Multicast (YAM) Routing Protocol*, Proceedings of HIPPARCH98, June 1998.

[14] K. Carlberg, J. Crowcroft, *A Comparison of Multicast Shared Trees Constructed with Different Metrics*, work in progress, 1998.

[15] M. Doar, *A Better Model for Generating Test Networks*, Proceedings of Global Internet, pp 88-93, IEEE, Nov, 1996

[16] Private discussion: J. Crowcroft, D. Cheritan, H. Holbrook, K. Carlberg, February 1999.

[17] D. Clark, Presentation at Munich IETF, July, 1997.

[18] D. Clark, Presentation at San Jose IETF, December, 1995.

[19] S. Corson, V. Park, *Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification*, Internet-Draft, Work In Progress, August, 1998.

[20] S. Corson, *MANET Routing Protocol Applicability Statement*, Internet-Draft, Work In Progress, December, 1998.

[21] J. Crawford, A.G. Waters, "Heuristics for ATM Multicast Routing", ATM '98 Sixth IFIP Workshop on Performance Modeling and Evaluation of ATM Networks, July, 1998.

[22] E. Crawley, et. al., *A Framework for Integrated Services and RSVP over ATM*, RFC-2382, August, 1998.

[23] E. Crawley, et. al., *A Framework for QoS-based Routing in the Internet*, RFC-2386, August, 1998.

[24] Y.K.Dalal, R.M. Metcalfe, *Reverse Path Forwarding of Broadcast Packets*, Communications of the ACM, 1040-1048, 1978.

[25] S. Deering, *Host Extensions for IP Multicasting*, RFC1112, August 1989.

[26] S. Deering, *Multicast Routing in a Datagram Internetwork*, Ph.D. thesis, Stanford University, 1991.

[27] S. Deering, et al., *An Architecture for Wide-Area Multicast Routing*, ACM Sigcomm'94 Conferencing Proceedings, October 1994.

[28] S. Deering, B. Cain, A. Thyagarajan, *Internet Group Management Protocol, Version* 3, Internet-Draft – Work In Progress, December 1997.
[29] S. Deering, Discussions on Mbone mailing list, February 1999.
[30] A. Dolan, J. Aldous, *Networks and Algorithms*, Wiley & Sons, 1993
[31] M. Donahoo, K. Calvert, E. Zegura, *Core Migration for Dynamic Multicast Routing*, Technical Report GIT-CC-95-28, March 1995.
[32] D. Estrin, et. al., *Protocol Independent Multicast - Sparse Mode: Protocol Specification*, RFC-2117, June 1997.
[33] D. Estrin, et al., *Protocol Independent Multicast - Dense Mode: Protocol Specification*, Internet-Draft work in progress, May 1997.
[34] D. Estrin, D. Meyer, D. Thaler, *Border Gateway Multicast Protocol (BGMP), Protocol Specification*, Internet Draft, Work In Progress, November, 1998.
[35] D. Estrin, M. Handley, A. Helmy, P. Huang, D. Thaler, *A Dynamic Bootstrap Mechanism for Rendezvous-based Multicast Routing*, Work In Progress, 1998.
[36] D. Estrin, et. al., *The Multicast Address Set Claim (MASC) Protocol*, Internet Draft, Work In Progress, September, 1998.
[37] H. Erikson, *MBONE: The Multicast Backbone*, Communications of the ACM, August, 1994.
[38] M. Faloutsos et. al., QoSMIC: Quality of Service sensitive Multicast Internet Protocol (QoSMIC), Proceedings of ACM SIGCOMM' 98, Sept, 1998.
[39] M. Faloutsos, Ph.D. Thesis, University of Toronto, November, 1998.
[40] D. Farinacci, et.al., *Multicast Source Discovery Protocol (MSDP*), Internet Draft, Work In Progress, June 29, 1998.
[41] D. Farinacci, A. Lin, T. Speakman, A. Tweedly, PGM Reliable Transport Protocol Specification, Internet Draft, Work In Progress, October, 1998
[42] P. Ferguson, *Simple Differential Services: IP TOS and Precedence, Delay Indication, and Drop Preference*, Internet-Draft – Work In Progress, November 1997.
[43] S. Floyd, et. al., A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing, IEEE Transactions on Networking, December, 1997.
[44] E. Gilbert, H. Pollak, *Steiner Minimal Trees*, SIAM Journal of Applied Physics, 1968.
[45] M. Garrett, M. Borden, *Interoperation of Controlled-Load Service and Guaranteed Service with ATM*, RFC-2381, August, 1998.
[46] Y. Goto, M. Ohta, K. Araki, *Path QoS Collection for Stable Hop-by-Hop QoS Routing*, Proceedings of INET'97.
[47] R. Guerin, S. Kamat, A. Orda, *QoS Routing Mechanisms and OSPF Extensions*, Internet-Draft Work In Progress, December 1998.
[48] Z. Haas, M. Pearlman, *The Zone Routing Protocol (ZRP) for Ad Hoc Networks*, Internet-Draft, Work In Progress, September, 1998.
[49] M. Handley, *An Examination of Mbone Performance*, USC/ISI Research Report:ISI/RR-97-450. http://www.aciri.org/mjh/mbone.ps.
[50] M. Handley, D. Thaler, *Multicast-Scope Zone Announcement Protocol (MZAP)*, Internet-Draft, Work In Progress, October, 1998.
[51] M. Handley, Ph.D. Thesis, University College London, 1997.
[52] M Handley, et. al, Hierarchical PIM, White Paper, December, 1997. ftp://cs.ucl.ac.uk/darpa/hpim.ps.gz
[53] S. Hanna, B. Patel, M. Shah, *Multicast Address Dynamic Client Allocation Protocol (MADCAP)*, Internet Draft, Work In Progress, October, 1998.
[54] C. Hedrick, *Routing Information Protocol*, RFC-1058, June 1988.
[55] S. Herzog, S. Shenker, D. Estric, *Sharing the "Cost" of Multicast Trees: An Axiomatic Analysis*, IEEE Transactions on Networking, December, 1997.
[56] H. Holbrook, public discussions on IDMR mailing list, February, 1998.

[57] F.P. Kelley, A.K. Maulloo, D.K.H. Tan, *Rate Control for Communication Networks: Shadow Prices, Proportional Fairness, and Stability*, Journal of the Operational Research Society, 1998, http://www.statslab.cam.ac.uk/~frank/PAPERS/rate.html
[58] S. Keshav, et. al., *A Framework for Differentiated Services*, Internet Draft, Work In Progress, October, 1998.
[59] J. Kurose, *Open Issues and Challenges in Providing Quality of Service Guarantees in High-Speed Networks*, ACM Computer Communications Review, 1993.
[60] B. Levine, J.J. Garcia-Lunes-Aceves, *Improving Internet Multicast with Routing Labels*, Proceedings IEEE Conference on Network Protocols, October 1997.
[61] A McAuley, M, Liu, R. Talpade, E. Bommaiah, *AMRoute: Adhoc Multicast Routing Protocol*, Internet-Draft, Work In Progress, August, 1998.
[62] Q. Ma, P. Steenkiste, *Routing Traffic with Quality-of-Service Guarantees in Integrated Services Networks*, Proceedings of NOSDAV'98, July, 1998.
[63] M. Macedonia, D. Brutzman, *MBONE Provides Audio and Video Across the Internet*, IEEE Computer, April 1994.
[64] T. Maufer, C. Semeria, *Introduction to IP Multicast Routing*, Internet-Draft – Work In Progress, October 1997
[65] S. McCanne, V. Jacobson, M. Vetterli, *Receiver Driven Layered Multicast*, Proceedings of the ACM, SIGCOMM'96, August, 1996.
[66] D. Meyer, *Some Issues for an Inter-Domain Multicast Routing Protocol*, Internet-Draft, Work In Progress, June 1997.
[67] D. Meyer, *Administratively Scoped IP Multicast*, RFC-2365, July 1998.
[68] D. Mills, *Network Time Protocol Specification*, RFC-1305, March, 1992
[69] D. Mills, *Exterior Gateway Protocol*, RFC-1584, March, 1994.
[70] P. Mockapetris, *Domain Names: Concepts and Facilities*, RFC-882, November, 1983.
[71] P. Mockapetris, *Domain Names – Implementation and Specification*, RFC-1035, Nov, 1987.
[72] J. Moy, *OSPF Version 2*, RFC-1247, July, 1991.
[73] J. Moy, *Multicast Extensions to OSPF*, RFC-1584, March 1994.
[74] K. Nichols, L. Zhang, V. Jacobson, *A Two-Bit Differentiated Services Architecture for the Internet*, Internet-Draft – Work in Progress, December 1997.
[75] Private conversation with A. O'Neil, BT Research.
[76] R, Pearlman, et. al., *The Simple Multicast Protocol*, Internet-Draft, Work In Progress, August, 1998.
[77] J. Postel, *User Datagram Protocol*, RFC-768, August, 1980.
[78] J. Postel, *Internet Protocol*, RFC-791, September 1981.
[79] J. Postel, *Transmission Control Protocol*, RFC-793, September, 1981.
[80] T. Pusateri, *Distance Vector Multicast Routing Protoco*l, Internet-Draft work in progress, February, 1997.
[81] Y. Rekhter, T. Li, *A Border Gateway Protocol 4 (BGP-4)*, RFC-1771, March, 1995.
[82] L. Rizzo, L. Vicisano, *A FEC based Reliable Multicast Protocol for Wireless Environments*, ACM Mobile Computing and Communications Review, April, 1998.
[83] Sanchez, et. al., *Quality of Service Extensions to OSPF or Quality of Service Open Path First Routing (QOSPF)*, Internet Draft, Work In Progress, September, 1997.
[84] H. Schulzrinne, S. Casner, R. Fredrick, V. Jacobson, *RTP: A Transport Protocol for Real-Time Applications*, RFC-1889, January 1996.
[85] H. Schulzrinne, *RTP Profile for Audio and Video Conferences with Minimal Control*, RFC-1890, January, 1996.
[86] H. Salama, I. Viniotis, D. Reeves, *Multicast Routing Algorithms for High Speed Networks*, unpublished memorandum, Jan. 1994, ftp://ftp.csc.ncsu.edu/pub/rtcomm/RTMulticast.ps
[87] S. Shenker, L. Breslau, *Two Issues in Reservation Establishment*, ACM Sigcomm'95 Conference Proceedings, October 1995.

[88] C. Shields, *The Ordered Core Based Tree Protocol*, Proceedings IEEE INFOCOM'97, April, 1997.
[89] H. Schulzrinne, S. Casner, R. Fredrick, V. Jacobson, *RTP a Transport Protocol for Real-Time Applications*, RFC-1889, January 1996.
[90] S. Shukla, E. Boyer, *Multicast Tree Construction in Network Topologies with Asymmetric Link Loads*, Naval Postgraduate School, Technical Report: NPS-EC-94-012, Dec, 1994.
[91] T. Smith, G. Armitage, *IP Broadcast over ATM Networks*, RFC-2226, October, 1997.
[92] M. Steenstrup, *Routing in Communications Networks*, Prentice-Hall, 1995.
[93] D. Thaler, C. Ravishankar, *Distributed Center-Location Algorithms: Proposals and Comparisons*, IEEE Infocom'96, March 1996.
[94] D. Thaler et.al, *The MASC/BGMP Architecture for Inter-Domain Multicast Routing*, Proceedings of ACM SIGCOMM' 98, Sept, 1998.
[95] D. Thaler, D. Estrin, D. Meyer, *Grand Unified Multicast*, Internet-Draft Work In Progress, August 1997.
[96] C. Topolcic, *Experimental Internet Stream Protocol: Version 2*, RFC-1190, October, 1990.
[97] P. Vixie, S. Thomson, Y. Rekhter, *Dynamic Updates in the Domain Name System*, RFC-2136, March 1996.
[98] Z. Wang, J. Crowcroft, *A Unified Framework for Multicast Forwarding*, Lecture Notes in Computer Science, Vol. 846, pp. 252-258, Springer-Verlag, 1994.
[99] Z. Wang, J. Crowcroft, *Quality of Service Routing for Supporting Multimedia Applications*, IEEE JSAC, Sept. 1996.
[100] Z. Wang, *User-Share Differentiation (USD): Scalable Bandwidth Allocation for Differentiated Services*, Internet-Draft – Work In Progress, December 1997.
[101] D. Waitzman, C. Partridge, S. Deering, *Distance Vector Multicast Routing Protocol*, RFC-1075, November , 1988.
[102] I. Wakeman, Ph.D. Thesis, University College London, 1995.
[103] D. Wall, *Mechanisms for Broadcast and Selective Broadcast*, Ph.D. Thesis, Stanford University, June, 1980.
[104] A.G. Waters, J. Crawford, "Low-cost ATM Multicast Routing with Constrained Delays", Technical report, University of Kent at Canterbury, Dec. 1997
[105] B. Waxman, *Routing of Multipoint Connections*, IEEE Selected Areas in Communications, December, 1988.
[106] L. Wei, D. Estrin, *Multicast Routing in Dense and Sparse Modes: Simulation Study of Tradeoffs and Dynamics*, IEEE Proceedings of the Conference on Computer Communications, June 1994.
[107] P. Whittle, *Optimization Under Constraints: Theory and Applications of Non-Linear Progamming*, Wiley & Sons, 1971.
[108] D. Zappala, D. Estrin, S. Shenker, *Alternate Path Routing and Pinning for Interdomain Multicast Routing*, USC CS Technical Report #97-655, June 1997.
[109] E. Zegura, K. Calvert, M. Donahoo, *A Quantitative Comparison of Graph-Based Models for Internet Topology*, IEEE Transactions on Networking, December 1997.
[110] L. Zhang, et. al, *RSVP: A New Resource Reservation Protocol*, IEEE Networks, Sept. 1993.
[111] L. Zhang, et. al., *A Framework for Use of RSVP with Diff-Serv Networks*, Internet Draft, Work In Progress, November, 1998.
[112] IDMR Working group Discussions of the IETF, June, 1997