The Use of IP-Anycast to Construct Efficient Multicast Trees

by

Dina Katabi

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1998

Author .....
Department of Electrical Engineering and Computer Science
September 30, 1998

Certified by .....
John T. Wroclawski
Research Scientist, Laboratory of Computer Science
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

ENG

The Use of IP-Anycast to Construct Efficient Multicast Trees

by

Dina Katabi

Submitted to the Department of Electrical Engineering and Computer Science
on September 30, 1998, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

This thesis consists of three parts: the design of a framework for an Internet-wide IP anycast service, the application of this framework in optimizing the cost of core based multicast trees, and the development of a new interdomain multicast protocol.

The anycast framework solves the intrinsic scalability problem associated with a global IP anycast service. It does so by dividing the interdomain anycast routing system into two modules. The first module provides low-cost anycast routes. The second module, run by the end domains, generates enhanced anycast routes that are customized according to the beneficiary domains interests. The design uses an adaptive caching system plus a route learning mechanism.

The second part of the thesis uses the IP anycast service to extend the Core-Based-Tree multicast protocol (CBT) with the option to build optimized cost dynamic-core trees. These trees are free from the two major drawbacks shown by CBTs – poor core placement and traffic concentration around the core router. The proposed extension makes all on-tree routers join the same anycast group such that the multicast tree appears to all off-tree routers as one network entity, which can be addressed directly. We use this property to direct data packets and join requests along the shortest path to the tree decreasing bandwidth consumption and join latency. Since the resulting tree is simultaneously an anycast group and a core based tree, we call it an ACBT. Our simulation shows that ACBT reduces bandwidth consumption by a factor of 25

In the last part of this thesis, we develop FleSc – a flexible and scalable inter-domain multicast protocol. Our protocol builds on the BGMP model to establish bi-directional trees rooted at an owner domain. However, our design differs from BGMP in two important ways. First, it uses ideas from our anycast framework to favor short local branches over hierarchical long ones. Second, the established tree uses hard state, which prevents transient loops and facilitates the establishment of alternative branches when the default ones do not accommodate the desired QoS. The architecture is scalable since it uses a small set of hierarchical aggregatable multicast routes that are extended at each end domain with a set of on-demand generated routes, which are customized according to the domains interests. It is flexible since it provides a variety of possible branches to chose from the one that reduces band-

width consumption or that satisfies certain QoS requirements. Moreover, it provides a mechanism to integrate such a branch in the tree without causing loops or starvation.

Thesis Supervisor: John T. Wroclawski
Title: Research Scientist, Laboratory of Computer Science

Acknowledgments

I would like to thank my thesis advisor, John Wroclawski, who provided advice and guidance throughout this work. His insightful comments and discussions have been very helpful in shaping the ideas in this thesis.

I am grateful to Dr. David Clark, Dr. Karen Sollins, and Dr. Mark Handley whose technical advice and feedback have been quite valuable.

Special Thanks are due to Wenjia Fang and Rob Cheng for making my graduate school an enjoyable and intellectually intriguing experience.

Thanks go to Xiaowei Yang and Le-wei Lehman for being inspiring colleagues and valued friends.

My thanks also go to the other members of the ANA group including Dorthy Curtis, Alex Snoeren, Rena Yang, and Elliot Schwartz who created a stimulating and enjoyable work environment.

Special thanks are due to my family.

Last, I would like to thank Marylin Pierce for all the extensions she gave to me until I finally handed in this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Thesis Overview

Through time, the Internet has evolved to surpass all the expectations of its founders. From a small simple network that provided connectivity between any two end points, the Internet grew to a huge region-based[1] inter-network that offers different types of connectivity and services. Today, the Internet provides point-to-point communication, point-to-multipoint communication, and multipoint-to-multipoint communication. It also supports services such as Internet Telephony, multiparty games, and Internet Conferencing. The current commercialized nature of the Internet is encouraging this trend, and pushing for even more services to open new markets. However, for the Internet to successfully continue developing new services, it needs a flexible and rich infrastructure that simultaneously facilitates the upper layer services and optimizes the consumption of network resources. An example of such an infrastructure is one that provides multicast routing, which facilitates multiparty communication while making efficient use of the available bandwidth. An even richer network infrastructure is one that also provides IP anycast, which enables a user to access the nearest of a group of receivers. Despite its usefulness, unfortunately this service is not yet well developed or understood.

By definition, IP anycast is a network service whereby a set of receivers that share the same characteristics are grouped together. A sender interested in communicating with a receiver with those characteristics contacts the anycast group, and the network arranges to direct the communication to the receiver nearest the sender. To give a feeling for the usefulness of anycast, we describe some of its applications. Probably the best known of them is locating the nearest of a set of replicated servers. For example, a set of replicated web servers would share the same

[1] We use the term region-based to describe the Internet topology, which is not exactly hierarchical, but rather looks like a set of interconnected autonomous regions that have well defined borders.

anycast address. A user interested in accessing a page on one of the servers directs his request to the shared address and the routers conspire to deliver the request to the closest server to the sender. Accessing the nearest server localizes the traffic, which maximizes user throughput [1], balances the load among the replicated servers [2], and saves network bandwidth. Anycast is also useful for dynamic host configuration. For example, by assigning the same anycast address to all Domain Name Servers, a newly booted host can use the anycast service to contact the local DNS. Source policy routing is another possible application whereby anycast is used, for example, to choose a long distance service provider. In this context, each long distance service provider configures all of his border routers with the same anycast address. To use long distance provider A, the sender chooses a route that has the anycast address of provider A as an intermediate point.

Because of its useful characteristics, when first introduced by RFC1456, anycast attracted a lot of attention. However, the early approaches were disappointing since the naive implementation of IP anycast consumes a routing entry for each global anycast group [3]. In this thesis we design a framework for IP anycast that retains its useful characteristics yet reduces the memory consumption considerably. Moreover, the design allows each domain to control the amount of memory dedicated to the service to optimize the obtained benefit.

Having provided a scalable IP anycast design, we explore the possibility of combining two network services to provide a more sophisticated one. Particularly, we use IP anycast to build more efficient shared multicast trees and solve the major problems associated with core based trees. To understand those problems one should review how Internet multicast evolved. In its early days, multicast routing was designed to use source rooted distribution trees [16]. Although such a mechanism minimizes the delay observed by the receivers, it consumes an unacceptable storage overhead and an excessive amount of bandwidth. Thus, shared tree multicast protocols were suggested to scale multicast to a large number of groups with multiple senders. Examples of such protocols are PIM-SM [21] and CBT [20]. Both protocols use explicit joining to a designated core or rendezvous point whereby receivers hear from all group sources, and sources reach all group members. By building shared trees, these protocols can achieve significant bandwidth and state savings, and improve scalability. However, building a shared tree around a core router introduces new problems. Since the core router attracts all of the traffic sent to a multicast group (to distribute it along the tree), it becomes inefficient when it is located away

from the set of receivers [28, 34]. Moreover, attracting traffic from all senders overloads the links around the core router and might cause congestion. This defeats the original purpose of shared trees, which is to use the same resources to accommodate a larger number of multicast groups [29].

Our solution to the above problems relies on the observation that both cases arise only for groups with high traffic rates. Groups with low traffic rates, even when mapped to the same core, do not cause congestion since the total traffic rate remains low. Also, the increase in bandwidth consumption caused by mapping these groups to distant cores is insignificant. Therefore, moving groups with high bandwidth requirements to smaller non-core trees solves the aforementioned problems.[2]

When using core-based trees, it is easy to locate the tree since it is tied to the core router. Both senders and receivers direct their packets and joins respectively towards the core. However, if non-core trees are used, then a new mechanism is needed to locate the tree. This is what anycast does. By assigning the same IP anycast address to all on-tree routers, not only can we locate a multicast tree, but we can also find the shortest path from a sender or a potential receiver to the tree. As such, data packets travel along the shortest path from a source to the tree consuming as little bandwidth as possible. Joins also make it to the tree along the shortest path, building short branches and consequently decreasing bandwidth consumption and join latency. Since the resulting tree is simultaneously an anycast group and a core based tree, we call it ACBT. Our simulation shows that ACBT reduces bandwidth consumption by a factor of 25%, and eliminates traffic concentration.

The real tradeoff when building an ACBT is between state and bandwidth since an ACBT consumes an additional routing entry to advertise the anycast address. However, not all multicast groups will use ACBTs, since CBTs are sufficient for groups with low traffic rate. Only groups with high bandwidth requirements will use ACBTs. Although the number of such groups is always limited by a domain's capacity, their impact is huge, since if mapped to the same core router, they can cause congestion and harm all the multicast and unicast traffic in the domain.

The last problem we tackle in this thesis is interdomain multicast. Currently, interdomain multicast is still immature. Most of the efforts are geared towards the Border Gateway multicast

[2] We use the term non-core tree for a shared tree that is not a minimum spanning tree rooted at a core.

Protocol (BGMP) developed by the Internet Engineering Task Force (IETF) [33]. However, since BGMP relies on MASC which is a hierarchical address allocation mechanism, it tends to favor longer hierarchical routes that connect two domains through their common antecedent provider over shortcut links that connect domains locally. Our interdomain protocol builds on the BGMP model to establish a bi-directional tree rooted at a root domain. However, our design differs from BGMP in two important ways. First, it uses ideas from our anycast framework to favor short local branches over hierarchical long ones. Second, the established tree uses hard state, which prevents transient loops and facilitates the establishment of alternative branches when the default ones do not accommodate the desired QoS.

The thesis is organized as follows: after an initial introduction, chapter two provides a summary of related anycast work. Chapter 3 describes the expected service by discussing the features and limitations of IP anycast and by comparing it to other approaches that support similar services. Chapter 4 explains the IP anycast framework. Chapter 5 gives a summary of related intradomain multicast work. Chapter 6 describes our extension to the CBT multicast routing protocol to provide it with an option to build ACBT – an optimized cost shared multicast tree. Chapter 7 presents FleSc, our interdomain multicast protocol. Finally, we conclude with a thesis summary and future work.

Chapter 2

Related Anycast Work

2.1 IP Anycast

The notion of anycasting was first introduced in Internet literature by RFC1546 [3]. The authors identified the need for an anycast service as an Internet-wide service location mechanism. They defined it as "a stateless best effort delivery of an anycast datagram to at least one host and preferably only one host, which serves the anycast address." The paper points out several architectural issues involved with IP anycast, such as how to integrate the IP's stateless service with the desire to have an anycast address represent a single virtual host, and what address space should be used to support anycast. The authors considered it "wiser to use a separate class" to assign anycast addresses as opposed to carving them from the existing unicast address space.

When the Internet Engineering Task Force began considering the idea of finding a replacement for IPv4, all major candidates adopted anycast as an addressing mode (Pip [7], SIPP [6], IPv6 [9]). The Simple Internet Protocol defines Cluster addresses as "unicast addresses that are used to reach the nearest one of the set of boundary routers of a cluster of nodes identified by a common prefix in the SIPP unicast routing hierarchy." In other words, a cluster address identifies a topological region in the Internet and has the same prefix shared by the nodes in this region. The proposed application for cluster addresses, when used as part of an address sequence, was to permit a node to select which of several service providers it wants to carry its traffic (policy source routing).

When SIPP evolved to IPv6 [9] the cluster addresses became the anycast addressing mode, which was redefined as follows: "An IPv6 anycast address is an address that is assigned to more than one interface (typically belonging to different nodes), with the property that a packet sent to

an anycast address is routed to the 'nearest' interface having that address, according to the routing protocols' measure of distance."

In contrast to RFC1546's recommendation of assigning anycast addresses their own address space, IPv6 specifies that anycast addresses are allocated from the unicast address space (the same as cluster addresses). It also specifies how to advertise those addresses: "For any assigned anycast address, there is a longest address prefix P that identifies the topological region in which all interfaces belonging to that anycast address reside. Within the region identified by P, each member of the anycast set must be advertised as a separate entry in the routing system (commonly referred to as a "host route"); outside the region identified by P, the anycast address may be aggregated into the routing advertisement for prefix P. Note that in, the worst case, the prefix P of an anycast set may be the null prefix, i.e., the members of the set may have no topological locality. In that case, the anycast address must be advertised as a separate routing entry throughout the entire Internet."

2.2 Application Layer Anycast

In this context, anycast is regarded as a communication paradigm whereby a user locates an Internet service. The exact details vary among proposals, but the basic idea relies on building a replicated server database that can be queried by clients interested in finding the shortest route between two Internet addresses. The information needed to answer clients' requests is collected from the routing system either directly or indirectly.

To give a flavor of application level anycast, we provide the following summary of two proposals:

- "The Host proximity service (HOPS)" proposed in [8] by Paul Francis provides, as its name indicates, proximity information to interested clients. The service relies on two types of systems, HOPS servers and HOPS probes. HOPS servers can calculate the distance between either two unicast addresses or a unicast address and the members of an anycast group. HOPS probes are responsible for collecting information about the network topology to populate the HOPS servers' database. To do so, each probe runs BGP with a nearby border router to determine the distance to some or all Internet address prefixes relative to itself. The

probes establish peer relationship with each other and with the HOPS servers and broadcast their information through a flooding mechanism similar to those used by link-state protocols. By collecting all the probes' views of the Internet, each HOPS server builds a high level picture of the topology that it uses to answer the queries it receives. The proposal does not state how the HOPS servers map an anycast group to the unicast addresses of its members, or how to keep such a mapping up-to-date.

- Ammar, Zegura, Fei, and Bhattacharjee propose in [1] an application-layer anycasting service that maps anycast domain names into IP addresses using anycast resolvers. The design assumes the existence of a hierarchical authority (similarly to DNS) that maps anycast names to their corresponding IP unicast addresses.[3] The paper does not discuss how the membership information should be collected, especially if the membership is dynamic.

  Clients are configured with the address of their local anycast resolver. When they need to access an anycast group, they query the local resolver with the anycast name and a metric to optimize. In case the resolver is authoritative for that service, or it has cached in its database the information to map the group to its members and the performance information to choose the member that optimizes the metric in the request, the resolver can answer the question. Otherwise, the local resolver determines the address of the authoritative resolver and obtains the anycast group information, which is thereafter cached in the local resolver. However, since many of the metrics, such as proximity, have only local meaning, it is not clear how the resolver should answer the request in this case. The paper suggests various possible metrics and different mechanisms to collect the metric information.

1.3 The Service Location Protocol

The Service Location Protocol (SLP), developed by the svloc group of the IETF, provides an automatic way for clients to discover services within their administrative domain [14]. The protocol involves three types of entities: Service Agents (SAs), which represent a service, a Directory Agent (DA), which handles a database holding information about all services in the

[3] It is not clear what the index is according to which this hierarchy is organized. It is also not clear that such a common index exists.

domain and their associated attributes, and User Agents (UAs), which represent clients looking for a particular service with a given list of attributes. The mechanism works as follows: All SAs and UAs have to discover the DA (or set of DAs) in their domain either by static configuration or by listening to a specific multicast group used by DAs to advertise themselves. SAs use a particular scheme to map their service and its attributes to a string, which they register with the local Directory Agent, which maintains a database for all the services in the domain. The UAs formulate their queries using the same scheme used by the SAs, and send them to the DA. The DA consults its database and replies with a set of servers that meet the requirements. In a domain that has no DA, UAs learn about SAs by using multicast queries to a predefined multicast address.

Chapter 3

Controversial Questions--The Service

Although the name of anycast is well known today in the Internet community, there is still a lot of confusion about the service anycast provides. Some of the controversial questions are the following: "does anycast solve the same problem as the Service Location Protocol?" "should anycast be provided at the network layer or the application layer?" "does anycast have other applications besides that of service location?"

3.1 Relation Between IP-Anycast and the Service Location Protocol

Two particular services are desirable in the future Internet. The first is a directory service that answers questions similar to: "What is the IP address of a telephony gateway that accepts credit card payment?" The system responsible for providing such a service should answer the client with a list of IP addresses that comply with the requested service and its attributes. If no server can support the requested service, the system replies with an empty list. To answer this question we need a language to specify the service, a mechanism by which servers register their services to advertise them throughout the entire Internet, and a mechanism that clients may use to access this information. This is the problem the service location protocol solves.

The second service is similar to the 800-number service, whereby a user who bought an account from company X can access any of company X's online offices, and preferably the one that provides him with the best performance. This is the problem that anycast tackles.

The essential difference between the two problems is that, in the first one, the user is the one to decide on which server to access according to his personal preferences (i.e., which one of the various telephony gateways to use? Sprint's or MCI's?) In the second case, the user has no information to choose between the different replicated servers, possibly because the choice is topology dependent, therefore he leaves the choice to the network, hoping for reduced cost and improved performance.[4] Thus, the two services described above are complementary and equally useful.

Currently the Service Location Protocol discovers services only in an Intranet or a domain, and has scaling limitations when applied to a wider area network. Even modifying the service location protocol later to improve scalability does not reduce the need for anycast, since the latter provides a different service.

3.2 The Advantages of IP-Anycast over Application-Layer Anycast

We suggest that anycast is better implemented at the network level for two reasons. First, while application layer anycast serves only for choosing among a set of replicated servers, IP anycast can provide many other important services such as policy source routing (whereby anycast is used to choose the long distance service provider), and ACBT (where the anycast service is exploited in optimizing the cost of multicast trees).

Second, even for the service location purpose, IP-anycast works better than application-layer-anycast. In this context, anycast is used to choose one of a group of servers that are all equal according to the user's knowledge. In other words, the user leaves the choice to the network because it does not have enough information to make the choice himself, otherwise he would have used something like the service location protocol. From the user's perspective, the best choice is the server that provides the user with the best performance (in terms of, for example, delay and throughput). From the network's perspective, the best choice is the one that consumes

[4] Note that theoretically the anycast metric can be anything: the number of hops, the level of congestion along the route, the server load. However, the experiment done in [1] shows that the number of hops correlates well with the throughput.

fewer resources, which is usually the closer server to the user. It is desirable to satisfy both the user and the network. It turns out that this is possible since an experiment done in [1] reveals that the number of hops separating a client from a replicated server (the simple IP-anycast metric) correlates well with the client perceived throughput. Figure 3-1 is taken from [1] and presents the results of the aforementioned experiment.

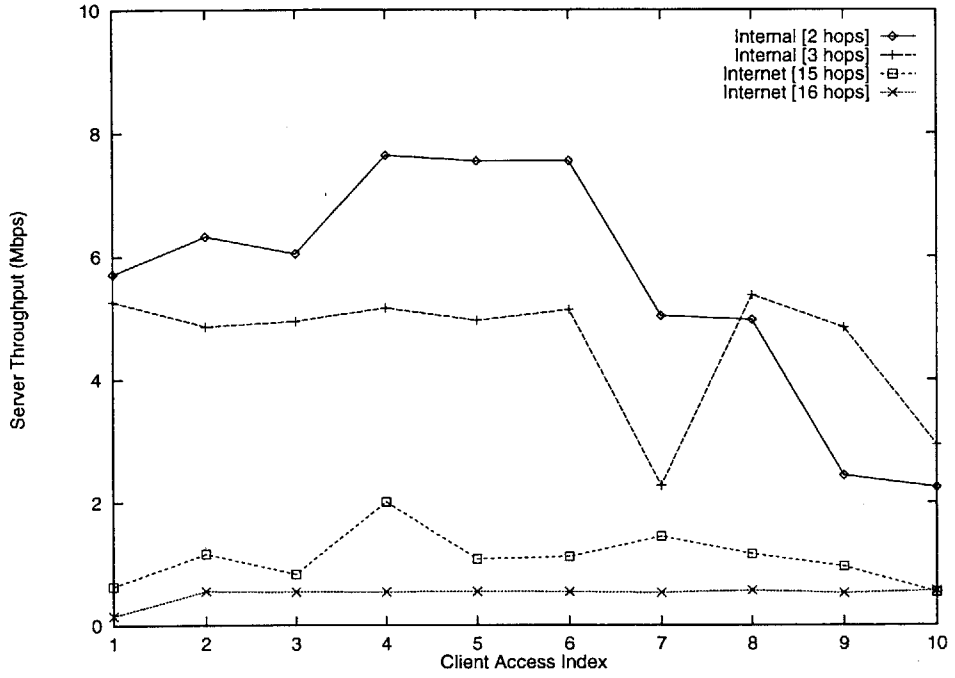


Figure 3-1: The effects of the distance measured by the numbed of hops on the server-to-user throughput. The x-axis indicates the index of the transfer and the y-axis gives the throughput (taken from [1]).

Moreover, application layer-anycast is composed of different components that all have scalability problems. To build a database that can choose among a set of replicated servers, we need a membership protocol and a mechanism to collect proximity information (or any other metric information according to which the system makes its choice). The membership protocol is used to decide which active server belongs to the anycast group at any point in time. How to collect such information about all servers in the Internet and how to keep it up-to-date are questions that none of the application-layer anycast proposals have answered. On the other hand,

collecting proximity information is a tough problem, since this data is of a distributed, volatile, and relative nature. It is volatile since it changes when a server or a link goes down, it is distributed since no single point has complete information about the Internet topology (or the load at each of the replicas, in case this is the metric of interest), and it is relative because a local server for a particular node is a distant one for another node.

Additionally, application-layer anycast resembles application-layer multicast because, to answer the proximity question, each anycast server has to keep information about all the servers in the Internet that belong to the anycast group. In other words, a nearby server does not hide the distant servers. In contrast, in IP-anycast the distributed nature of the system provides each router with exactly the information it needs to know, no more and no less.

Chapter 4

A Global IP Anycast Framework

4.1 Design Goals

We want IP anycast to meet the following goals:

1. The service should be Internet-wide.
2. The service should be scalable, which implies that the overhead in terms of storage, traffic and processing is manageable in a large heterogeneous inter-network.
3. The service should be efficient. This means that the service locates the nearest member of an anycast group with high probability, and that accuracy increases as the distance between the user and the nearest member of the addressed anycast group decreases.
4. The service should fit the increasingly commercialized nature of the current Internet. As such, the incurred cost has to be located in the same domain as its associated benefits.
5. The service should not make the Internet less secure than it currently is.

4.2 Assumptions/Predictions

We design an IP anycast framework for an environment where the following assumptions (predictions) are true[5]:

1. Most of the anycast groups will show a high temporal and spatial locality that justifies the use of some caching mechanism. Precisely, at any given time there is a predictable set of services

that hosts in domain D will access with a high probability, and the more popular a service is at a domain D, the higher the probability that it is offered somewhere close to D. Although no statistics exist at this stage regarding interdomain anycast traffic, unicast inter-AS route statistics reveal high temporal & spatial locality [40]. We expect inter-domain anycast traffic to show even more temporal and spatial locality than its unicast counterpart since most of the anycast applications are of the service-location type, where a client attempts to locate the nearest of a group of servers. In this context, it is likely that the users' demands correlate well with the replicated servers' distribution, because an organization providing an Internet service will attempt to well provision its service by distributing its replicated servers according to its customers' demands. Note that locality of traffic is both a reality and a necessity, which any architecture should encourage.

2. There is a market for a host anycast service, and there are going to be companies offering services through the Internet that would like to equip their customers' software with their anycast address, and get them to use it to access their services anywhere. The willingness of these companies to pay for such a service will push ISPs to increase gradually their network ability to handle anycast. Thus, we want a design that lets the impact of an improvement in the ISPs network infrastructure be reflected smoothly on the anycast service.
3. We expect the anycast topology to be more dynamic than the unicast topology. In particular, we expect intradomain anycast to be dynamic since it can be used for nomadic computation and process migration.[6]
4. Service providers are willing to provide some network resources at the borders of their networks to support anycast.

4.3 Architectural Principles

Our design emphasizes the following concepts in a high-level architecture that addresses routing and addressing issues.

---

[5] In this thesis, we do not discuss how the system degrades when the assumptions become invalid.

[6] An attractive application of intradomain anycast is Intranet-server-migration. In an Intranet environment traffic statistics are very dynamic and traffic measurement devices are available. Thus, to balance traffic, one would think of migrating a database server to where it is mostly accessed.

- In the early days of SIPP[7], Steve Deering suggested cluster addresses. A cluster is a set of connected nodes that all hold the same IP address. Deering's view of a cluster was to represent an ISP's network, such that a user can choose its long-distance service provider by using this provider's cluster address as an intermediate address in its source routed packet. At present, this view of anycast as a method to access virtual network structures has faded, and anycast has been primarily considered a service location mechanism. We do think that it is important for IP anycast to capture both perceptions in a general flexible framework. Thus, we perceive IP-anycast as a region access method where a region could be a node, a set of connected nodes, a set of separate nodes, or any other network entity that has an IP address[8].
- The Internet is getting more commercialized every day. In such an environment it is much easier to price a service when the cost is located in the domain where the service is sold. For example, if ISP A wants to provide his customers with an IP anycast service most of the incurred overhead should be handled by ISP A (or its parent provider).
- The basic service of the Internet is to provide global unicast connectivity, which should always be available to every machine connected to the Internet. The nature of this service is different from other services in that it is the basic building block that other services build upon.

4.4 Design Description

Our architecture distinguishes between internal and external anycast groups. An internal anycast group for domain D is a group for which D has at least one member. Internal anycast groups are routed via unicast, which means that each member is advertised as a separate entry in the routing system, and that each router in domain D knows the nearest anycast member. The number of internal anycast groups is limited (Assumption1) and administratively controlled.

An external anycast group for domain D is a group for which D has no members. The basic objective of our anycast framework is to solve the scalability problem inherent in routing external

[7] Simple Internet Protocol Plus.
[8] Chapter 6 builds on this idea and uses anycast as a method to access a multicast tree.

anycast groups. To do so we rely on two mechanisms: caching and the use of already available unicast topology information. Our solution follows the argument below:

There exist two known mechanisms to reduce the state information in a system: hierarchical aggregation and caching. Anycast groups are not amenable to hierarchical aggregation since a group might have members anywhere and it might even migrate over time. However, we think that anycast is amenable to caching since clients in a domain will tend to access the same services (groups) repeatedly. We call the set of anycast groups that clients in a domain keep accessing the popular groups for that domain. Thus, a domain does not need to know the shortest route to all anycast groups in the Internet; it only needs to know the routes to the popular groups (those popular in that domain). Routes to unpopular anycast groups are seldom (if ever) used and should not establish state in this domain.

The problem with the above argument is that routing in the Internet is done hop-by-hop, or in the interdomain context, routing is done domain-by-domain. In other words, if domain A is connected to B which is connected to C, then for C to forward to A, domain B must know how to forward to A. Thus, if G is an anycast group in A, and G is popular in domain C, C will not know about the existence of a member from G in A unless B transfers this information to it. More important, C can't send to G unless B keeps information about the existence of G in A. This is because for C to send to G it has to go through B. However, if B does not keep state for how to get to G it will drop C's packets on the floor. The conclusion is that the naive forwarding mechanism requires every domain to keep state not only for groups that are popular in that domain but also for all groups that are popular in the downstream domains.

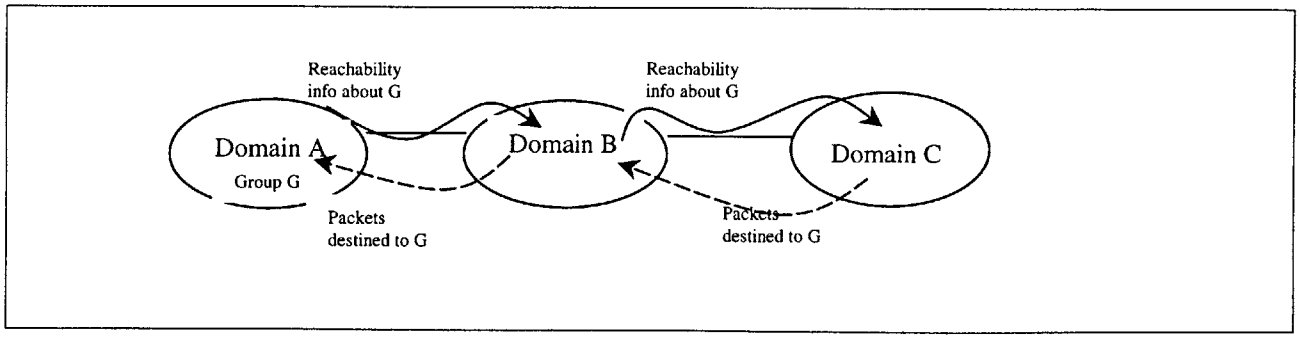


Figure 4-1: Routing anycast a la unicast

To overcome this shortcoming we need two mechanisms. First, C should learn about the existence of G in A. Second, C should encapsulate the anycast packet in something B knows how to route to domain A. So let us assume for the time being that C learns about the routes to the nearest members of its popular groups, and that C knows how to map each group to a unicast address so that other domains that do not know the anycast route still can forward the packet. The last problem we have to deal with is how to route to unpopular anycast groups. Occasionally clients in domain C generate packets destined to unpopular groups for which C has no route. In this case domain C should have a fallback mechanism that may not discover the nearest member, yet it finds a member of the anycast group that can provide the service. The question is what fallback mechanism to use.

Our design requires every anycast group to choose an address that indicates the unicast prefix of a domain that has at least one member of the anycast group. As such, when the domain has no cached information about the anycast group it forwards the packet toward the unicast prefix using its unicast routing table. Other domains in the path do the same until the packet gets to the domain whose prefix is indicated by the anycast address. For this domain the anycast group is internal, thus every router knows how to route to the nearest anycast member.

In the next sections we explain how a domain learns the routes to popular groups, how to map a group to a unicast address, and how the anycast address of a group indicates the owner domain.

Let us again review the mechanism: Border routers keep track of the popular groups in a domain. They initiate the learning mechanism to find the route to the nearest member of each popular group. When an anycast packet is addressed to an external group, it is routed to a border router. If the BR has a cached route for that packet, then it sends the packet according to the cached route, otherwise the BR sends the packet according to the fallback mechanism towards its owner domain.

Thus, our design is a hybrid of the IPv6 anycast mode and RFC1456's global IP-anycast. It is similar to IPv6 in that with each anycast group, it associates a region in which the group is internal and fully advertised; however, it does not confine the group to that region. Members can exist outside the owner region and still receive the anycast packets. It is similar to the global anycast service described in RFC1456 in that packets are delivered to the nearest anycast

member anywhere in the Internet; however, it does not require every router to have information about every anycast group in the entire Internet.

The next three sections give detailed description of the design.

4.4.1 Address Architecture

The problem with the previous IP-anycast proposals, is the legendary belief that IP anycast will be routed exactly as IP unicast. However, since the anycast addresses do not obey the ISP hierarchy, the naive unicast routing mechanism would result in each anycast address being present in every router's memory. Such a solution obviously does not scale to a large number of global anycast groups.

Since we want to provide a global service, we conclude that anycast can't be routed the same as unicast, and that anycast addresses should be differentiable from their unicast counterparts. Therefore, we assign anycast addresses their own address space, but we allocate them according to the unicast hierarchy. As such, an anycast address is a concatenation of the anycast indicator,[9] the unicast prefix of the owner domain, which assigned the address, and the group ID. Thus, a router receiving an anycast packet can in the worst case shift off the anycast indicator and forward the packet to the owner domain using the unicast routing table.

4.4.2 Address Assignment

We mentioned in the previous section that an anycast address is a concatenation of the anycast indicator, the unicast prefix of the owner domain, and finally the group ID. The fact that the anycast address is a unicast address shifted to the right and concatenated with the anycast indicator implies that each domain owns an anycast address space that is proportional to its unicast address space.

Each domain is responsible for assigning anycast addresses from its anycast address space. For example, if domain D has a unicast prefix P, and if the anycast indicator is X, then domain D's anycast prefix is XP.

[9] The anycast indicator is a bit pattern that identifies the class of anycast addresses.

The process according to which a domain D assigns an anycast address to an end user is domain dependent and can be the same as the one used for assigning unicast addresses. For example, the anycast address might be assigned manually by the administrator, or the domain might have special address assignment servers that lease anycast and unicast addresses with certain lifetime to end-users.

The design requires the user to setup at least one machine with the anycast address in the owner domain. However, the user can assign the address to other machines anywhere in the Internet.

4.4.3 Address Advertisement

Anycast reachability information is generated and propagated only by routers. A host that wants to join an anycast group has to ask its next hop router to advertise the address on its behalf, which can be achieved by adding a new message type to either IGMP or the neighbor discovery protocol. A router that receives such a request processes it through some security checking procedure, and if compliant it marks the address to be advertised periodically according to the anycast routing protocol adopted by the domain. The router uses a keepalive mechanism to ascertain the availability of the anycast member. It never advertises an address after the member becomes inaccessible.

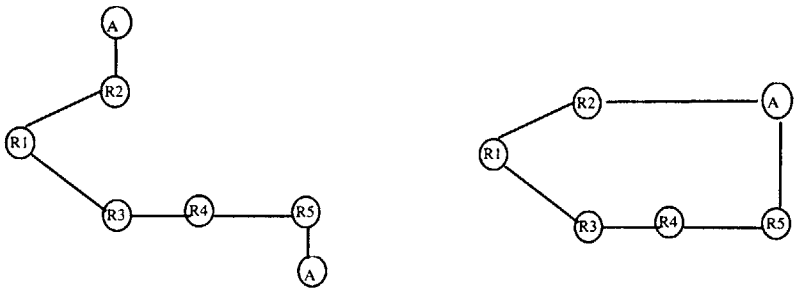
4.4.4 Routing

4.4.4.1 Intradomain Anycast Routing (Routing Internal Groups)

An internal anycast group for domain D is a group for which D has at least one member. Internal anycast groups are routed via unicast, which means that each member is advertised as a separate entry in the routing system, and each router knows the nearest anycast member.

The same algorithms that are used for unicast routing are also used for anycast routing. The Distance Vector algorithm works without any modification. For the Link-State algorithm to work correctly, routers should abstain from routing through an anycast address. For example, router R1

in Figure 4-2-a should not mistake the topology as that in Figure 4-2-b and should not try to route packets sent to R5 through A. This problem is easily fixed since in our framework routers can differentiate between an anycast and a unicast address. To solve the problem, the router assigns a large cost to virtual links connecting anycast nodes to their local networks, such that they are not used in building routes unless the anycast node is the destination.



(a) (b)

Figure 4-2: Applying the link state algorithm directly on the topology may introduce false topologies

4.4.4.2 Interdomain Anycast Routing (Routing External Groups)

Routing anycast at the interdomain level is the real challenge. If we do it similarly to unicast routing, we end up spreading information about every anycast group in the entire Internet. This would consume even more space than a unicast table because anycast addresses do not obey the ISP hierarchy and do not make any use of Classless Inter-Domain Routing [12].

An external anycast group for domain D is a group for which D has no members. External groups are either popular or unpopular. We call the external groups that clients in a domain repeatedly access the popular groups for that domain. Identifying the popular groups is a function of the border routers (BRs), which keep track of the anycast addresses accessed and periodically check this information to decide on the popular anycast group.

External groups are routed using an interdomain anycast routing protocol run by border routers. This protocol assumes that all BRs in a domain share the same anycast address, which we call the domain's anycast address (DAA). The protocol is built on top of TCP similarly to

BGP, and uses the same mechanisms used by BGP to unify the BRs view of the topology. According to this protocol, the BRs in a domain keep track of the anycast addresses of the popular groups in that domain. For each popular group, the BRs initiate a search mechanism to learn about the nearest domain that has a member of that group. If the search is successful the BRs cache the learned route and inject it in every subsequent anycast packet that is addressed to the group. If a BR (or any router) receives an anycast packet for which no cached entry exists the BR shifts the anycast indicator in the destination address and route the packet according to its unicast forwarding table. Assuming that clients in a domain are interested in contacting a small subset of the total number of anycast groups, the number of anycast routes will be small and manageable. Also the spatial locality assumption implies that the learning mechanism will take little time and will most likely be successful.

- **The search mechanism**

The search mechanism is a domain-by-domain broadcast that collects a vector of domains secluding this domain from a neighboring domain that has a member of the anycast. The search mechanism is triggered by the exit BR towards the owner domain, which receives the anycast packets in the absence of a cached route. This BR, which we call the responsible BR for the anycast group (RBR), contacts all of its internal peers asking them to start the search procedure. Each of the internal peers sends a search message to its external peers that includes the RBR's address and a counter set to the longest route the domain will cache measured in number of domains. This should be around 3 or 4 since most stub domains are closer than 4 domain-hops to the backbone [42]. The message is delivered according to the rules BGP uses to forward its route advertisements (over TCP connections and complying with the same rules to which BGP messages comply when traveling between external and internal peers.) When a search message is received from an external peer that has no information about the anycast group, the receiving BR adds its domain's AS-number to the route vector and decreases the search counter by one. If the counter reaches zero the message is discarded. The BR propagates the message to all of its internal peers, which forward it further to their external peers. If the receiver BR knows that its domain has members of the anycast group, it stops the broadcast, and unicasts the reply back to the RBR after adding its domain's AS-number and DAA. Also, if the BR has a cached entry for that anycast address, it concatenates its entry with the route vector, decreases the counter

according to the length of its path to the nearest member it knows about, and if the counter is still greater than zero it sends the message back to the RBR.

After sending the search message the RBR sets a timer and waits for replies. When the timer expires, the RBR compares the replies it has received so far. The RBR selects the shortest route giving priority to routes that are in the direction of the owner domain. The RBR advertises the learned route to its internal peers as if it were learned through BGP. If the RBR fails to find a route, it decides it is most likely that there is no nearby members of that anycast group, and it schedules another trial later (possibly with an exponential backoff.)

A cached anycast route contains the domain vector, which describes the set of domains the route traverses, and the anycast address of the destination domain (DAA). Both are extracted from the search reply. The vector is kept to answer search messages issued by neighboring domains looking for a route to this anycast group. The anycast address of the destination domain (DAA) is injected in all subsequent anycast packets sent to this group [10].

- **Maintaining a cached Route**

A learned route becomes invalid in the following cases. First, when the originating domain loses connectivity to the destination domain. In this case, the RBR discovers the loss of connectivity from its unicast routing table (Note that the DAA indicates the destination domain's unicast prefix) and initiates a new search. The second case happens when the nearest anycast member crashes or leaves the group. In this case the originating domain can't discover the lost route directly and keeps using the route to send anycast packets. However, when those packets get to the destination domain, the receiving BR discovers that there is no local anycast entry. Thus, it forwards the packets to the owner domain, and sends an ICMP message towards the source informing it of the invalidity of the learned route. The ICMP message is tagged with a special tag that is captured by the BRs of the source domain. A BR that receives such an ICMP contacts the RBR for the group, which treats the message similarly to a route withdrawal received via BGP and conducts a new search.

Note that if at any time a new anycast member appears in a domain that anycast packets cross on their way to their owner domain, the anycast packets are delivered to this nearer member.

[10] Either via encapsulation or source routing.

However, this feature does not occur when the anycast packets are sent according to a learned entry; therefore, the domain has to conduct periodic searches to ensure that no closer member has joined the anycast group since the last search. However, because anycast servers do not tend to join and leave a service often, this periodic search could be done on a coarse time granularity. It is also possible to use a push mechanism. In this case, we require every domain to inform its neighbors every time an external anycast group becomes internal. This happens when the first member in the domain joins an anycast group that is not owned by the domain. The message is hopped domain by domain, and includes a counter to restrict it from going to distant domains (this is equivalent to a localized BGP advertisement.)[11]

- **Forwarding an Anycast Packet**

Thus, an internal router receiving an anycast packet either has an entry for the anycast address in its routing table (group is internal) or it does not (group is external). In the latter case, it shifts the anycast indicator and uses its unicast table to send the packet towards the exit BR for this anycast group. If the RBR has a cached entry for the address it injects in the packet (by source routing the packet to the learned DAA.) Otherwise, the RBR shifts the anycast indicator and forwards the packet towards the owner domain.

All routers in the path between the originating domain and the destination domain forward the packet according to their unicast tables (no matter whether the packet is routed using the cached entry or the default route.) When the packet is received by its owner domain or by the destination domain of a cached route, the receiving border router delivers the packet to its internal component which forwards it to the nearest member of the anycast group.

4.5 Discussion and Evaluation

Are the design's goals met?

The architecture provides an Internet-wide IP anycast service. Anycast group members are accessed by their neighbor clients even when they are outside the owner domain.

[11] However, intermediate domains that are not interested in the group do not have to store it in their table.

The overhead is manageable since the internal routers in a domain know only about internal groups, whose number is limited and administratively controlled. Border routers maintain cached routes for popular external anycast groups whose number is also limited by the clients' interests. BRs conduct search procedures to learn about nearby popular groups; however, the searches are done on a large time granularity and are likely to succeed quickly given our assumptions.

Only domains that have group members internally carry the overhead of advertising. These domains can charge for the advertising service. The search mechanism is fully carried out by the interested domain, which can police the accessed groups and charge its clients for providing an anycast service.

The service is efficient. If the anycast group is internal to the domain the service locates the nearest member. If the group is external and popular the service also locates the nearest anycast member in terms of domains traversed. Note that search messages could handle information about the traversed domains' size to enable an originating BR to prefer a route that crosses small domains to a route that crosses wide domains. In case of unpopular anycast groups the packets are delivered to the nearest anycast member on the way toward the owner domain. Though the packets in this case are not necessarily delivered to the nearest member, this event happens only occasionally and does not degrade the service in general.

What are the problems that this design solves and others do not?

There are two known approaches for an IP anycast service. The first is described in RFC 1456, whereby each global anycast group is advertised (via unicast routing) to all routers in the Internet. Such an approach has dramatic scalability problems that prevent its applicability.

The second approach is the IPv6 approach, which allocates anycast addresses from the unicast address space. As such the anycast addresses owned by a domain are no different from its unicast addresses and are fully advertised inside the domain. Outside the owner domain the anycast addresses are aggregated with their unicast counterparts into the domain's prefix. Thus, members of the anycast group outside the owner domain are not visible. As a result, to provide a global service one should acquire a prefix of null and advertise the address throughout the entire Internet, which does not scale. In particular, the IPv6 architecture attempts to provide scalability by confining each anycast group to a priori defined topological region. However, previous

experience in Internet addressing shows that such an approach does not succeed since it runs the risk of limiting the future expansion of the service. A case similar to that faced by class-B addresses will arise. All users will tend to choose an anycast address whose scope is larger than the scope where they offer their service to avoid the possibility of an address change in the future, e.g. in case the company grows bigger.

Another drawback of the IPv6 approach comes from its dependence on unicast routing. The CIDR architecture used for aggregating unicast routing entries relies on the ability of discovering a longest match between one of the forwarding table entries and the forwarded packet destination. Though this architecture achieves good aggregation, it complicates the forwarding process by enforcing the use of complex data structures and search/insertion mechanisms. In the case of anycast, the longest match concept is meaningless and not used. However, if anycast addresses are no different from unicast addresses then anycast routes are forced to suffer the unnecessary cost of complicated algorithms and data-structures that were optimized for finding the longest match.

In contrast, our design is a hybrid of the IPv6 anycast mode and RFC1456's global IP-anycast. It is similar to IPv6 in that with each anycast group it associates a region, in which the group is internal and fully advertised; however, it does not confine the group to that region. Members outside the owner region are visible in their neighborhood and attract the anycast packets. It is similar to the global anycast service described in RFC1456 in that packets are delivered to the nearest anycast member anywhere in the Internet; however, it does not require every router to have information about every anycast group in the entire Internet.

Chapter 5

Related Intra-domain Multicast Work & Motivation

5.1 Algorithms for Calculating Multicast Trees

Early IP multicast protocols were based on Deering's work [16] and solely used source specific shortest path trees. According to this design, a multicast group builds multiple distribution trees, each rooted at one of the group's sources. Although these protocols minimized the delay perceived by the receivers, they consumed an unacceptable amount of storage overhead and an excessive amount of bandwidth. As multicast became more popular, and the number of groups larger it became obvious that source-specific-tree routing protocols, which spread per sender state, can't scale to the wide area network and definitely not to the entire Internet. The solution was to build one tree per group that is shared among all senders. Although this approach helps reduce the state overhead, it increases the delay seen by the receivers. On the other hand, the bandwidth used by a shared tree depends on the type of tree we choose to build, and could be less or greater than a source-specific tree. Ideally, we would like to build a shared tree that minimizes the bandwidth consumed to deliver traffic to all of the group members. However, building the minimal cost shared tree is an NP-complete problem widely known as the Steiner Minimal Tree (SMT). SMT minimizes the bandwidth necessary to deliver traffic to all receivers, yet the computational complexity prohibits its use in a networking environment.

To cope with the complexity of SMT, Wall suggested the use of core-based-tree algorithms [17]. A core-based–tree, as the name indicates, uses a shortest path tree rooted at a core or a rendezvous point. The tree is shared, and all senders address their traffic to the core router, which forwards the packets along the tree edges. Although finding the core that minimizes the cost of the

tree is an NP-complete problem, it is possible to build the tree around a suboptimal core and achieve acceptable performance. Actually, a core-based tree rooted at one of the receivers consumes bandwidth comparable to an SBT (Source Based Tree) and has slightly higher average delay. However, these trees show high traffic concentration especially when the same core is used for multiple groups. In case of high traffic rate, the heavily loaded links around the core become bottlenecks degrading the network's ability to handle more traffic, even further than would be the case if SBTs were used [29].

Doar & Leslie studied a different type of shared trees [32]. They compared the performance of what they call "the naïve multicast algorithm" with that of a KMB tree, which is an approximation of SMT that has an average cost 5% more than SMT. The naïve multicast algorithm computes the multicast route by combining the shortest path across initial multicast group's members, then joining new members to the nearest attachment point on the tree. Their results showed that the cost of the naïve tree is within 1.5 times that of the KMB trees, and that the maximum delay is around 50% to 60% of that of KMB trees. The problem with this algorithm is that it needs global knowledge of the network topology, which is usually not available.

5.2 Intradomain Multicast Routing Protocols

Intradomain multicast routing protocols can be classified into three types according to how the multicast tree is established: broadcast and prune (e.g., DVMRP, PIM-DM), membership advertisement (e.g., MOSPF) and rendezvous-based (e.g., CBT, PIM-SM). DVMRP and PIM-DM broadcast initial data packets which trigger and store prune states in those parts of the network that do not lead to group members. MOSPF broadcasts membership information so that the intermediate nodes can construct source specific distribution trees on the fly.

In contrast, Core Based Trees [20] and Protocol Independent Multicast-Sparse Mode [21] do not use broadcast. Rather, they use explicit joining to a designated core or rendezvous point whereby receivers hear from all group sources, and sources reach all group members. By building core-based shared trees, these protocols achieve considerable bandwidth and state savings, and improve scalability. However, they increase bandwidth consumption and traffic concentration.

5.2.1 The Core Based Tree Multicast Routing Protocol (CBT)

The Core Based Multicast protocol constructs shared bi-directional trees rooted at a core router. To join a tree a receiver multicasts an IGMP host membership report on its attached link. This report causes a local CBT router to invoke the tree join process by generating a JOIN_REQUEST that travels towards the core. Upon receiving the join message, a router sets up a transient join state, and forwards the join to the next node towards the core. The join travels upstream until it hits an on-tree router or the tree's core, which then generates a JOIN_ACK. The JOIN_ACK traverses the reverse path of the corresponding join message guided by the transient state. When the JOIN_ACK arrives at the originating router, the branch is fully established.

It is worth noting two characteristics of CBT. First, the tree established is bi-directional, which means that traffic can flow in both directions along any branch. Second, CBT is a hard state tree. In particular, once the tree is built it is never reshaped unless one of the on-tree links fails, in which case the branch downstream from the failed link is flushed and reestablished.

5.3 Motivation: What is wrong with CBTs?

Core-based trees exhibit two major deficiencies[12]. The first is traffic concentration around the core, which is caused by senders addressing their packets to the core of the tree causing a high traffic concentration around the core router and a threat of congestion. The problem is exacerbated for groups with high traffic rate especially when they are mapped to the same core router. Note that core routers are usually located in the center of their domain, which makes them critical waypoints for both multicast and unicast traffic. As such, their congestion harms all sort of traffic in a domain: video streams, telnet, ftp, and many others.

The second problem is the possibility of a poor core placement. Since the core constitutes the only point of distribution of traffic to receivers, its ideal location would be in "the middle" of the

[12] This is true regardless of whether they are PIM-SM type or CBT type.

corresponding group's members. A core that resides away from its group's members causes the multicast tree to grow unnecessarily larger, consuming more bandwidth and storage space.

Chapter 6

Intra-domain ACBT

ACBT is not a multicast routing protocol, it is rather an optional feature that customizes a core-based-tree to accommodate groups with high traffic rate and localized topology.

ACBT is designed to work in a domain that supports both Anycast and CBT-Multicast. However, since intra-domain anycast does not require any special protocol, and can rely solely on the available unicast routing protocol,[13] ACBT works correctly in a domain that does not support anycast explicitly.

6.1 Architecture

ACBT is engineered to solve the two major problems exhibited by core-based trees, which are "distant core placement" and "traffic concentration" around the core router. The solution relies on the observation that both problems arise only for groups with high traffic rate. Groups with low traffic rate, even when mapped to the same core, do not cause congestion since the total traffic rate remains low. Also, the increase in bandwidth consumption caused by mapping these groups to distant cores is insignificant. Therefore, moving groups with high bandwidth requirements to smaller non-core trees solves both problems.

Our design relies on the concept that anycast is a network access mechanism which allows us to access a virtual region inside the network using the shortest path. Since a multicast tree can be viewed as a virtual region then by assigning the same anycast address to all on-tree routers the whole tree appears to any off-tree router as one network entity; the multicast group can be addressed directly from any router in the domain. This allows packets destined to a multicast

group to travel along the shortest path connecting the source to the nearest on-tree router, which minimizes the off-tree part of their journey. Joins also travel along shortest paths from a potential member to any on-tree router, therefore they build short branches and reduce the consumed bandwidth. The resultant tree approaches a minimum-spanning tree, which considerably reduces the total bandwidth needed to deliver the traffic to all members. Moreover, the access point to a multicast tree is no longer tied to the core router. Each sender sees the nearest on-tree router as the access point to the tree, which prevents traffic concentration on any particular on-tree link. Finally, the tree becomes free to move inside the network and is not tied to a particular core router.

6.2 Design Details

6.2.1 Building an ACBT

Integrating the ACBT option in the Core Based Tree protocol is easy, all we need to do is to use the group anycast address wherever CBT uses the core address. When an ACBT is created, the initiator requests both a multicast and an anycast address for the new group. After acquiring the needed addresses the initiator uses an out of band mechanism to advertise the new group to its potential members (the anycast address is included as part of the group's description).

In contrast to CBT, where non-member senders encapsulate their packets and unicast them to the core, in ACBT they encapsulate them in IP packets destined to the associated anycast address. The intervening routers conspire to deliver them to the nearest on-tree router, which forwards them along the multicast tree.

To join an ACBT, a host multicasts on its attached link a join request, which contains both the group address and the associated anycast address.[14] On receiving this request, a local ACBT aware router invokes the tree joining process (unless it has already) which will follow one of two possible techniques. In case the router does not find an entry for the anycast address in its routing

[13] See Chapter 4.

[14] The host needs a mechanism to inform its next-hop router about the mapping between the group it desires to join and its RP represented here by the associated anycast address. This can be done by defining a new message type for either IGMP or the Neighbor Discovery Protocol.

table, it infers that it is the first member to join this multicast group and it has the responsibility to initiate the associated anycast group. Therefore, the router follows a special serialization mechanism to ensure that no more than one router succeeds in starting the same ACBT (see 6.2.2.)

In case the joining router finds an entry for the associated anycast address, it forwards a JOIN_REQUEST message (the one defined in CBT' s specs) to the next hop on the path towards the anycast group -- the "target-router" field is set to the anycast address. This join message must be explicitly acknowledged (JOIN_ACKed) by any router that holds the anycast address, usually the nearest on-tree router to the host. The join message sets up a transient join state in the routers it traverses, which consists of <group, previous hop, next hop>. "Previous hop" is taken from the incoming control packet's IP source address, and "next hop" is gleaned from the routing table -- the next hop to the specified anycast address.[15] This transient state eventually times out unless it is "confirmed" by a join acknowledgment (JOIN_ACK) from upstream. The JOIN_ACK traverses the reverse path of the corresponding join message, which is possible due to the presence of the transient join state. The JOIN_ACK causes each router that has a transient state for the corresponding group to set a timer whose expiration causes the router to start advertising itself as a member of the anycast group. The timer filters out tree fluctuations and unstable branches. Once the acknowledgment reaches the router that originated the join message, the new receiver can receive traffic sent to the group.

To prove that ACBT is loop-free, we assume that the existing tree does not contain a loop, and prove informally that a newly added branch does not create a loop. According to the above design, in ACBT, all on-tree routers simulate the role of the core. In other words, the whole multicast tree shows the same input and output as a huge core, which has edges connected to all routers adjacent to the tree. Thus, the establishment of a new branch in an ACBT simulates building the first branch in a CBT. Therefore, if CBT is loop-free then ACBT is also loop-free.[16]

---

[15] We do not discuss the case where the router is on a multicast link, which is similar to CBT.

[16] This should not be understood as skepticism of CBTs correctness. Actually, though we are not aware of a formal prove of correctness, CBTv2 is simple enough to see that no fundamental loops exist.

6.2.2 Serializing the initializations of an ACBT

It might happen that multiple routers attempt to initiate the same ACBT group at the same time. Thus, we need a mechanism that ensures no more than one of these routers succeeds in its attempt. Any serialization mechanism can solve the problem. For example, a centralized serialization method would work (a domain might have a special serialization server that every initiating router should contact before starting an ACBT. The server issues an initialization permit as long as no permit was given before for initiating that ACBT.)

The solution we recommend is the following: The initiating router multicasts to all-cbt-routers an ACBT_INIT message informing them about its intention in starting this multicast/anycast group. The router waits for a period greater[17] than twice the domain's RTT, if it does not hear from any other CBT router and still has no entry in its routing table for this anycast address, the router decides that it is the first on-tree router, declares itself the root for the multicast tree, and advertises itself as a member of the corresponding anycast group. If after multicasting the ACBT_INIT message, the router hears an ACBT_INIT multicast for the same group sent by a router with a smaller IP address, or if the router hears an objection, it fails in its attempt.

A router should unicast an objection to the originator of an ACBT_INIT multicast in either of the following situations: the router has a smaller IP address, and has itself sent an ACBT_INIT multicast for the same ACBT group, or the router is the root of an ACBT that has the same multicast address. In this case the root sends two objection packets narrowly spaced.

The mechanism described above favors the router with a smaller IP address. If the timer is set appropriately, the mechanism only fails when two packets are lost (the two multicasts, or the multicast and the objection sent by the router with the smaller IP address, or the two objections sent by a root router.) One could strengthen the mechanism by asking for an objection-ack or by multicasting the initialization multiple times. However, even if the mechanism fails and two routers decide to be roots for the same multicast group, our design provides a mechanism for them to discover the multicast forest and merge it into one tree or flush one of the trees and have its members rejoin.

6.2.3 Pruning

An ACBT is pruned both downstream-to-upstream and upstream-to-downstream. The former happens whenever the child interface list of a non-root router becomes NULL, in which case the router sends a QUIT_NOTIFICATION to its parent. The latter occurs when a root that has no directly attached members loses all of its children except one. In this case the root is no longer needed to connect the tree and can resign leaving its responsibilities to its only child. To do so, the root leaves the anycast group, sends to its child a ROOT_QUIT message, and waits for a ROOT_QUIT_ACK. If the root does not receive a ROOT_QUIT_ACK within a certain interval, it resends the ROOT_QUIT message. The root keeps trying to resign as long as it thinks it is connected to its child. Similarly to CBT, the root discovers loss of connectivity from the absence of echo messages and reacts by flushing the tree [20].

When the root receives an acknowledgment from its child it deletes all of the tree information and stops replying to echo messages sent by the child. A child that receives a ROOT_QUIT from its parent replies by a ROOT_QUIT_ACK and declares itself a root. This mechanism helps the tree to move flexibly in the domain and never get tied to a particular node.

If the ROOT_QUIT message is lost, then the only difference is that the root is no longer part of the anycast address and does not attract additional joins. This is desirable behavior since the root is preparing for resignation and is not a necessary part of the tree. If the ROOT_QUIT_ACK message is lost then the child will stop sending echo message and data packets to the root. The root will keep attempting to resign until it receives an acknowledgment, or until it infers loss of connectivity from the absence of echo messages.

If the root receives a JOIN_REQUEST after it has resigned, it can't acknowledge the join, but must process it towards the anycast group. Received by the root's child, the join implicitly carries a ROOT_QUIT message, which the child acks if it has not done so already. Even if the join is directed toward an on-tree router different from the child to which the root sent a ROOT_QUIT message and the join gets acked, this will not cause loops. Thus, although the child that was supposed to become a root might have lost the ROOT_QUIT message and still be sending multicast traffic to what it thinks is the root, the resigned root does not accept such

[17] How much greater than 2*RTT depends on the underling unicast /anycast routing protocol. It will be smaller for

packets from its previous child because it only accepts multicast packets from links it considers to be on the tree.

Note that a root can't quit while in the middle of a merge (see next section,) nor it can start a merge after it has sent a ROOT_QUIT message.

6.2.4 Maintenance

In ACBT, as in CBT, the parent-child relation is established according to the direction the founding join has traveled. Tree maintenance is also similar to the way it is performed in CBT. The only difference is that a flush message causes the receiver to leave the corresponding anycast group, before it propagates the flush downstream.

More precisely, each on-tree router, except the root, is responsible for maintaining its upstream link, provided it has interested downstream receivers. This maintenance is achieved by sending periodic ECHO_REQUEST messages to the parent. The receipt of such a message over a valid child interface prompts an on-tree router to send immediately an ECHO_REPLY downstream. Although ECHO_REQUEST messages do not include group information, ECHO_REPLY messages periodically inform the downstream router about the groups for which the upstream router is its parent.

If a downstream router does not receive an ECHO_REPLY from its parent after [GROUP_EXPIRE_TIME] it infers loss of connectivity with the parent, leaves the corresponding anycast group, and flushes its down stream subtree. Each router that receives a FLUSH message from its parent replies by a FLUSH_ACK upstream, leaves the anycast group, and sends a FLUSH message downstream. When the flush is received by a router that has directly connected receivers, the router schedules a new join to be sent after a timeout. The timer ensures that the loss of connectivity that caused the flush has been propagated to routers inside the domain, and that the routing system had enough time to converge. This eliminates the possibility of a join looping inside the network because of transient loops in the underling unicast system. Note that this timer is necessary for both ACBT and CBT, however, in ACBT it plays an additional role, which is to drain the stale anycast information from the routing system. This task should not take

---

link-state protocols.

more time than the propagation of the original change in connectivity that caused this subtree to be flushed.

6.2.5 Role of the Root

The root is the only node on a tree that has no parent, and as such it identifies the tree. Note however, that a root does not identify a multicast group because a group might have multiple unconnected trees due to network partitions.

The root performs all duties handled by a CBT core except acting as a rendezvous point for the group. Also, the root is responsible for any maintenance or control protocols that might be added in the future to CBT (such as Domain Wide Reports). Moreover, a root router is responsible for advertising itself to downstream routers, and to ensure that there is only one distribution tree for the group.

Every on-tree router knows the IP address of the tree's root. This information is learned through the JOIN_ACK the router receives when it first gets on the tree. If at any point later the root resigns, the new root has to inform every router on the tree of the change. The root also has to make sure that there is only one tree for the multicast group. To accomplish this task, a root router needs a mechanism to discover other instances of the multicast tree, and a mechanism to transform a multicast forest into one tree. Discovering other trees for the same multicast group can be achieved for free in a domain that supports Domain-Wide Reports. Precisely, a root router joins the group on which cores and root routers report their groups. If it hears its group reported by another router, it concludes that there is another tree rooted at the reporting router for the same multicast address. If the reporting router unicast address is lower than this root unicast address, this root is responsible for transforming the multicast forest into one tree. If the reporting router has a higher unicast address, the root router multicasts a Domain-Wide report message announcing the same multicast group.

In a domain that does not support Domain-Wide Reports, every root router multicasts periodically to all-cbt-routers a control message advertising its group. These messages are suppressed if the group gets announced by another router whose unicast address is lower than this root's address. When a root router discovers another root for the same group whose unicast

address is lower than its address it starts the process of transforming the forest into one tree rooted at the root with lower IP address.

Transforming the forest into one tree rooted at the root with lower IP address can be achieved in two ways. The simple way would be for the root with the higher IP address to flush its downstream tree causing the downstream receivers to rejoin the group. The second method would be for the root with the higher IP address to join the root with the lower IP address as if it were joining a core router. The question is which one to choose the Flushing method or the Merging method?

Merging is difficult; it requires every on-tree router to know for sure the root of its tree. Otherwise a joining root might cross one of its downstream routers causing a permanent loop. Flushing, on the other hand, is simpler and does not require downstream routers to know the identity of the root router. However, flushing causes a period of loss of connectivity that might be unacceptable for some applications.

In a networking environment, a multicast forest most likely is a result of recovering from a network partition. Network partitions are isolated events and are usually caused by one insubstitutable link going down. In such a context the tree built by merging the two trees has an efficiency comparable to one built after flushing. However, we still have to deal with the additional complexity caused by the need to inform every on-tree router of the IP address of the root. Yet, every on-tree router receives this information in the JOIN_ACK when it first gets attached to the tree, and this information stays valid as long as the root does not resign. Thus, a root knows whether it was always a root or it has become a root because its parent has resigned at some point during the tree's life. When a root discovers that it should join another root for the same multicast tree, it checks this information. If the root got the position from a resigned parent it flushes the tree letting every receiver rejoin on its own. If the root was always a root for the multicast tree it attempts to merge the two trees.

The merging process requires the root router whose address is higher to send a ROOT_JOIN_ REQUEST to the next hop on the path towards the root with a lower unicast address, to be referred to hereafter as the preferred root. On receiving this message, routers that do not have an entry for this group establish a transient state for the group and propagate the message one hop up-stream towards the preferred root. Routers that do have an entry for this multicast group,

check their tables, and if their corresponding root matches the preferred root they acknowledge the join back causing the transient branch to be confirmed. If an on-tree router receives a ROOT_JOIN_REQUEST and upon checking its table finds that the root of its tree differs from the preferred root, it quits its parent (unless the parent is the next hop towards the preferred root,) sets up a transient state for the join, then propagates the ROOT_JOIN_REQUEST one hop closer to the preferred root. Eventually the ROOT_JOIN_REQUEST will either hit an on-tree router whose root is the preferred one or it will be received by the preferred root itself, and in both cases the receiving router acknowledges the request. Once a router receives a ROOT_JOIN_ACK for which it has a transient state it confirms the state, propagates the ack down-stream, and if it has not done it already, it starts advertising itself as a member of the associated anycast group. On the other hand, if a ROOT_JOIN_REQUEST times out at any non-root router which has children then the router should send a flush message to all of its children, and stop advertising itself as a member of the corresponding anycast group.

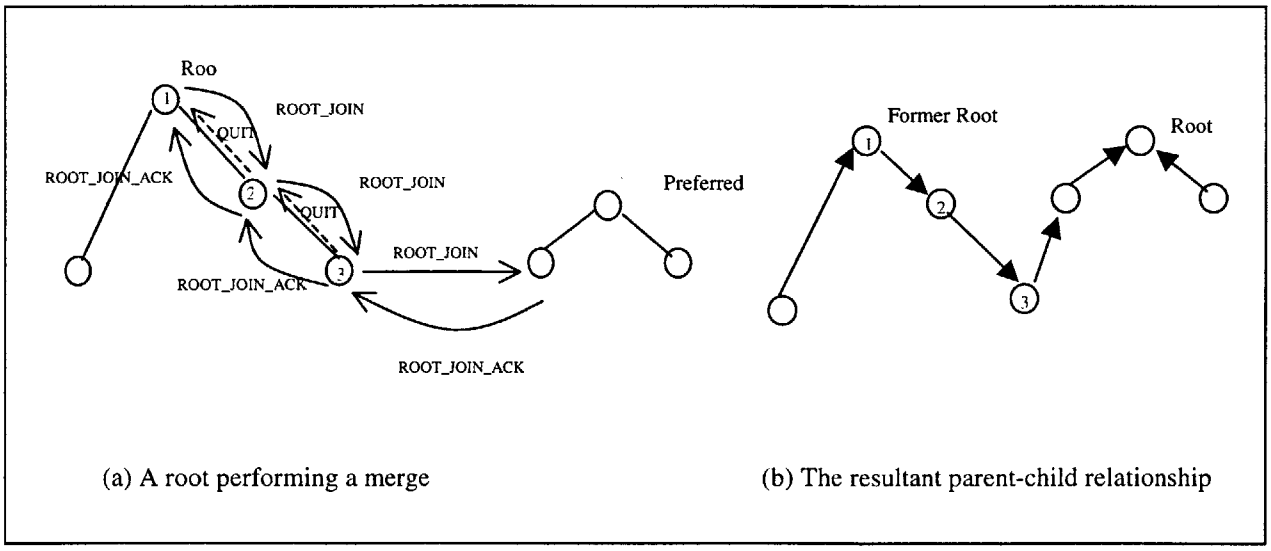


Figure 6-1: An illustration of the merge process.

Loops can't occur in this context since only routers on the tree rooted at the preferred root can ack the ROOT_JOIN. Starvation also can't occur since a ROOT_JOIN message causes any router that it crosses to propagate it towards the preferred root.

Obviously, one would like a root router not to leave the multicast group very soon after multicasting a Domain-Wide Report. However, if during the time a root router was trying to merge two trees the preferred router were to leave the multicast group, then the join will timeout and the root router attempting the merge will flush its downstream tree.

6.2.6 Forwarding Data packets

An on-tree router forwards a multicast packet over all the tree edges except the one it has received the packet on. An on-tree router does not accept multicast traffic delivered over an off-tree link.

An on-tree router accepts all anycast packets addressed to the associated anycast group, decapsulates them and propagates them over all its on-tree interfaces.

6.3 Discussion and Evaluation

This section is devoted to analyzing the design's overheads and benefits, and comparing them with those encountered by shared and source-specific trees.

- **Overhead**

The state overhead

The first cost incurred by the design is the state storage needed. For the mechanism to work correctly and efficiently, routers in the domain have to route the supporting anycast address and grant it an entry in their tables. Thus, each ACBT in a domain incurs an additional piece of storage in every router in that domain. However, the total number of ACBTs in a domain should be small since ACBT is designed for groups with high traffic rate, whose number is limited by the domain's capacity. Moreover, ACBT is an intradomain multicast distribution tree, and

internal routers in a domain usually do not suffer shortage of memory. Furthermore, the anycast address associated with a multicast group is no more than a suggestion provided by the members to enable the establishment of a better distribution tree. The final decision to build an ACBT depends on the initiating router, which can fall back to a CBT if it notices a shortage of memory.[18]

On the other hand, the alternatives do not use less state overhead. Source specific trees require per source information for every group. PIM-SM uses a source specific mode in which a designated router decides to switch to an SBT for a particular source because the source rate exceeds certain threshold, and takes no consideration of other factors such as the group topology and the number of sources, and the traffic burstiness. When receivers are located far from senders and senders send long bursts of traffic, this mechanism ends up spreading source-specific state throughout in the whole domain. Moreover, for a designated router to be able to switch to an SBT it has to monitor traffic from all sources for which it has no source specific state, this requires a lot of storage and processing overhead at the designated router. A shared tree protocol with a Bootstrap mechanism eliminates traffic concentration. However, it does not necessary use less state overhead since it requires all routers inside the domain to keep up-to-date information about all RP-candidates regardless of the number of active multicast groups.

The control traffic

We do not think that control traffic constitutes any significant overhead on the system. First Domain-Wide Reports are needed anyway. Second, advertising an associated anycast address is done through routing updates whose periodic generation is not affected by the existence or absence of the anycast entry.[19]

The delay in initializing the multicast tree

The serialization mechanism introduces a small delay in initializing a multicast tree. However, this delay constitutes a negligible fraction of the initialization latency, which is mostly caused by the multicast address allocation and advertisement. Moreover, the join data collected in [35]

[18] The router still has to go through the serialization process to ensure that no ACBT exists for this group.

reveals a long left tail and that users tend to join hours before the beginning of a multicast session.

The Usage of anycast addresses

The last overhead incurred by the design is the use of both a multicast and an anycast address per group. However, this mechanism is meant to be used with IPv6 where addresses are abundant. Moreover, we do not expect the number of ACBT groups to be huge since ACBT is most appropriate for groups with high bandwidth requirements and the number of such groups is limited by the domain's capacity.

- **Advantages**

Reduction in the consumed bandwidth

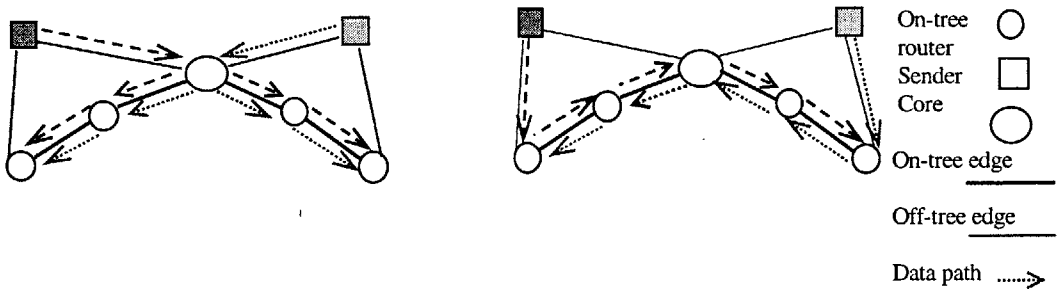
The major benefit in using ACBT is minimizing the overall consumed bandwidth. ACBT attempts always to build the shortest possible branch and as such it approaches a minimum spanning tree. Its bandwidth consumption on average is much better than both source-specific trees and other shared trees. This characteristic is particularly important for applications that consume a large amount of bandwidth such as video sessions.

No traffic concentration

The most difficult problem that faces shared tree protocols is traffic concentration at a core or rendezvous point. The problem can happen at two levels. First, different sources send simultaneously to the same group and cause congestion at the core. Second, multiple multicast groups are mapped to the same core causing their trees unnecessarily to share some links around the core and consequently causing their packets to be delivered over the same links. ACBTs alleviate traffic concentration at both levels. To illustrate this point, consider the situation in Figure 6-2; both Sender1 and Sender2 are sending video flows to a group of receivers. In 6-1-a

[19] For protocols that do not generate periodic updates the anycast updates would generate some extra traffic. However, given the small size of an update and the small number of ACBT groups, and the average time a receiver spends on a tree [35] we still think that the anycast updates' bandwidth consumption is negligible.

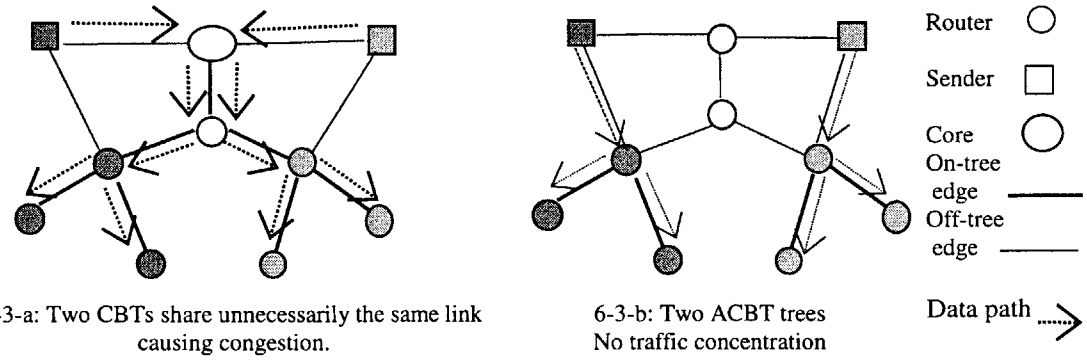
the two streams share the same physical links causing congestion, however when an ACBT tree is used the two flows use different physical links. In Figure 6-3-a two groups with high bandwidth requirements are mapped to the same core router and overload one of the links around the core. Figure 6-3-b shows the same setting with ACBTs, which achieve a more balanced traffic distribution.



6-2-a CBT with 2 video senders
The two streams share the same links

6-2-b ACBT with 2 video senders
the two streams travel over different physical links

Figure 6-2: ACBT reduces traffic concentration caused by multiple senders sending to the sam group simultaneously



6-3-a: Two CBTs share unnecessarily the same link causing congestion.

6-3-b: Two ACBT trees
No traffic concentration

Figure 6-3: ACBT reduces traffic concentration caused by mapping multiple groups to the same core router

Reduction in average delay and join latency

ACBT reduces average delay and join latency since packets travel along the shortest path connecting a source to a tree and get delivered over a small size tree with short branches.

Data ordering

Another advantage of ACBTs over source-specific trees used by PIM-SM is that they preserve data ordering. Consider the case in PIM where receivers are senders, and assume that M1's data rate caused M2 and M3 to switch to a source-specific tree for M1, yet M2 still uses the shared tree to receive packets from M3. It is possible for M3 to multicast a packet that triggers a response from M1, but M2 receives M1's response before M2's packet since the former is sent on a SBT and the latter uses an SHT.

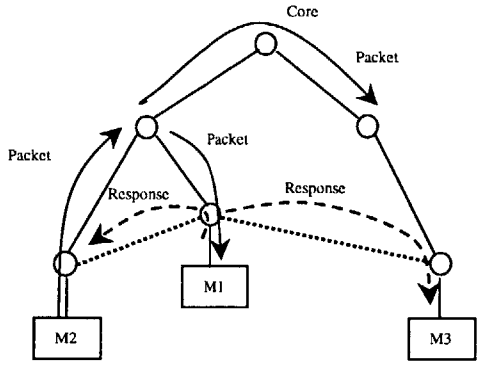


Figure 6-4: Data inconsistency in PIM-SM

Robustness

Finally, we claim that our design is more robust than other shared tree mechanisms. This robustness follows from the distributed nature of the scheme. First, there is no single point of failure, as it is the case for other shared tree protocols where a core going down causes hundred of groups to lose connectivity and all receivers in those groups to simultaneously seek reconnection and endanger the stability of the domain. Second, ACBT trees do not rely on a BSR (Bootstrap Router) to locate their groups, and consequently do not have to fear a mistake made by a BSR that harms all the multicast sessions in the domain. Senders do not need to be

connected to a Bootstrap enabled router to send to a multicast group, not even to a multicast or anycast enabled router. Senders can easily send to an ACBT located outside their domain. Moreover ACBT trees can soon recover after a network partition, where it takes other shared trees hours to do so [24]. And if ever multiple trees get constructed for the same group, ACBT discovers it and merges those trees together without disturbing the communication going on each of them.

Fanout control

A light control of the fanout is possible in ACBT through leaving the anycast group, which would prevent the router from attracting additional join-requests. However, the router might still receive joins if it is on the shortest path to other on-tree routers.

6.4 Simulation

6.4.1 Simulation Goals

Our simulation has the following goals:

- Quantify the potential benefit in a likely network environment.
- Understand the different parameters that affect ACBTs performance.
- Identify the characteristics of multicast groups for which ACBT is most suitable and the characteristics of domains whose administrator should enable the ACBT option.

6.4.2 Performance Metrics

- Cost

This metric reflects the total bandwidth used by the group to carry the traffic from senders to receivers. In other words, it measures both the on-tree consumed bandwidth, and the off-tree

bandwidth consumed by encapsulated multicast packets. We estimate the cost by counting the total number of times a queue in the simulation processes a data packet.[20]

- Traffic Concentration

Traffic concentration is measured by the number of flows traversing each unidirectional link in our simulation. Both the maximum number of flows on a link and the distribution of this number are meaningful.

- Delay

Traditionally multicast trees simulations investigated only the propagation delay. In real life, pathological delay cases are always caused by queuing delay and retransmission and propagation delay is rarely significant. We think that ACBT with its ability to balance the links' load can eliminate many of the long delay cases. However, following the tradition we simulate both the average propagation delay and maximum propagation delay and compare them to those of CBT. We define both average delay and maximum delay from the receiver perspective:

1. Average delay: the sum of the delay seen by each receiver for every packet over the total number of the received packets.
2. Maximum delay: the maximum delay seen by all receivers and for all packets.

6.4.3 Simulation Environment

- Topology: all of the graphs used in the simulation are generated using the Georgia Tech ITM topology generator [47]. We used the Doar-Leslie edge connection method and the Waxman edge connection method to generate 50-node and 100-node flat graphs. We also created transit-stub graphs to test ACBT performance in hierarchical topologies.
- We modified Network Simulator (ns) to simulate the CBT multicast routing protocol. The CBT module performs the functionality listed in CBTv2's specifications.
- A static anycast routing is used in the simulation. Such simplification holds the implicit assumption that the unicast/anycast routing protocol works at finer time granularity than the

[20] All packets are the same size, and all links have the same capacity.

Join/Leave dynamics. We think this is a justified assumption given the statistics in [35], which state that the average join interarrival is in the range 7-70 minutes.

- In every simulation, all senders (CBR sources) start sending simultaneously. Receivers join at different times previous to the senders' start time. Senders are stopped a few minutes later, then the simulation is stopped after delivering all packets in transit. Senders and receivers are chosen randomly among the nodes in the domain. We simulate the same groups first with ACBTs then with CBTs.
- In a flat domains we choose the core to be the node with the highest edge degree [27]. In a hierarchical domain we choose the core to be the transit node with the highest edge degree. As such we are simulating the best behavior CBT can practically achieve. However, if RPs are assigned randomly to groups (the Bootstrap mechanism) then CBT performs much worse [20].

6.4.4 Simulation Results

A Case Study

We used a 50-node graph generated using the Doar-Leslie edge connection method. The graph has an average edge degree of 3 and a bicomp of 11 [47]. Groups' size is chosen to be 10% the domain size for the first set of simulations, and 20% of the domain size for the second set of simulations. We run 50 different simulations in each set to measure the average delay, the maximum delay and the tree's cost for both CBT and ACBT. Figure 6-5 below shows that the ACBT average cost is 25% less than its CBT counterpart. Average and maximum delay are also reduced.

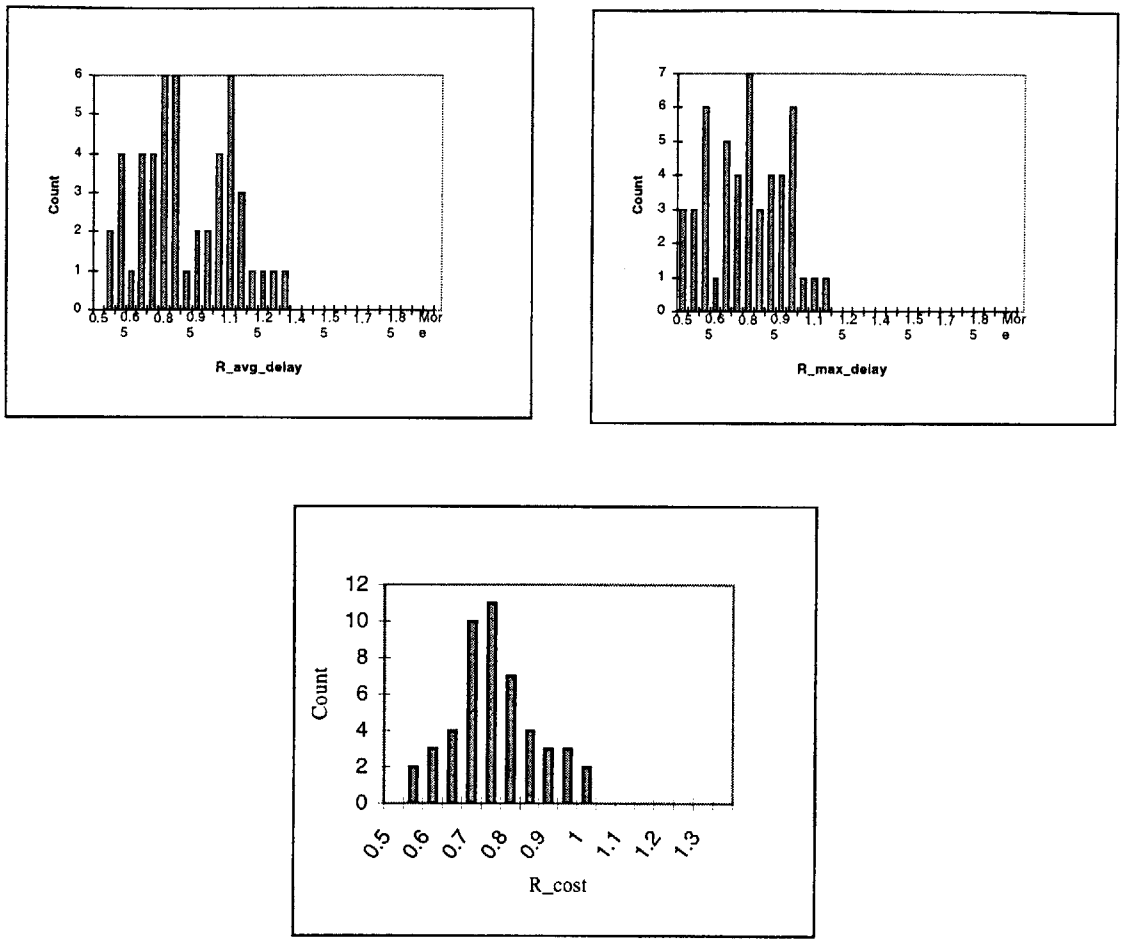
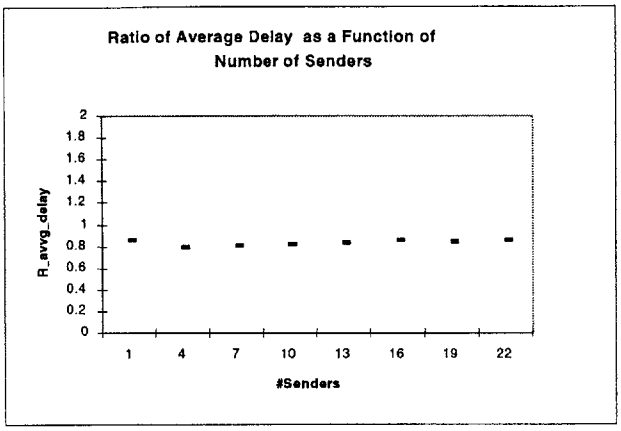
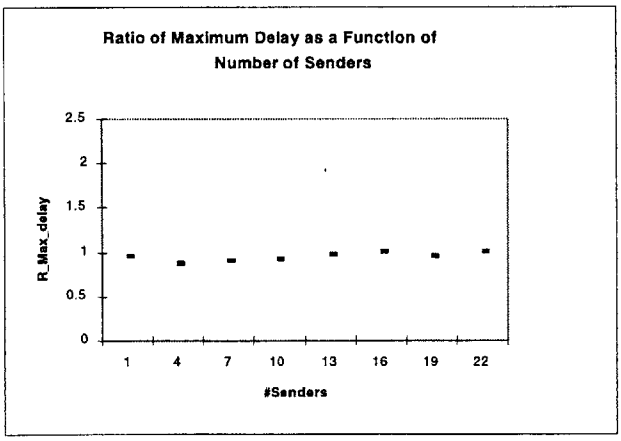


Figure 6-5: The ratio of average delay, maximum delay, and cost in ACBT to their counterparts in CBT.

Delay and Cost as Functions of the Number of Senders

Figure 6-6 shows that the number of senders does not affect the cost and delay ratios. Note, however, that the total cost (as opposed to the ratio) increases with the number of senders in both CBT and ACBT. This means that a 25% decrease in cost refers to a larger absolute decrease in bandwidth when the number of senders is greater. Therefore, groups that have many senders where each of them sends a considerable amount of traffic are encouraged to use ACBTs.





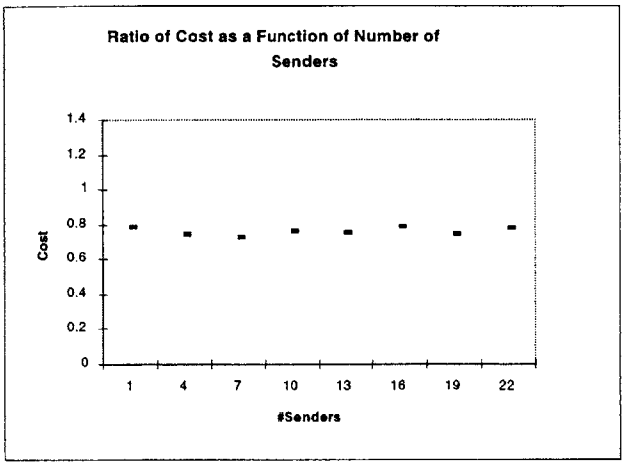


Figure 6-6: The ratio of average delay, maximum delay, and cost in ACBT to their counterparts in CBT as functions of the number of senders.

Delay and Cost as Functions of the Number of Receivers

Table 6-1 shows that increasing the group size increases the delay and cost ratios. This increase is expected since the groups are randomly generated which means that they have no localized topology. Thus, ACBT performs efficiently with groups whose size is less than 20% of the domain's size. However, ACBT is efficient for larger groups with localized topology.

Membership density	R_avg_delay	R_max_delay	R_costs
6%	0.861867	0.965955	0.718653
12%	0.944777	1.079844	0.768105
18%	0.867785	0.937148	0.79866
24%	1.060556	1.154418	0.791684
30%	1.012112	1.090173	0.824746
36%	1.134483	1.203614	0.833631
42%	1.26041	1.299737	0.856148
48%	1.256647	1.255669	0.875222
54%	1.163343	1.244954	0.883335
60%	1.191496	1.25877	0.913455

Table 6.1: The ratio of average delay, maximum delay, and cost in ACBT to their counterparts in CBT as functions of the group's density.

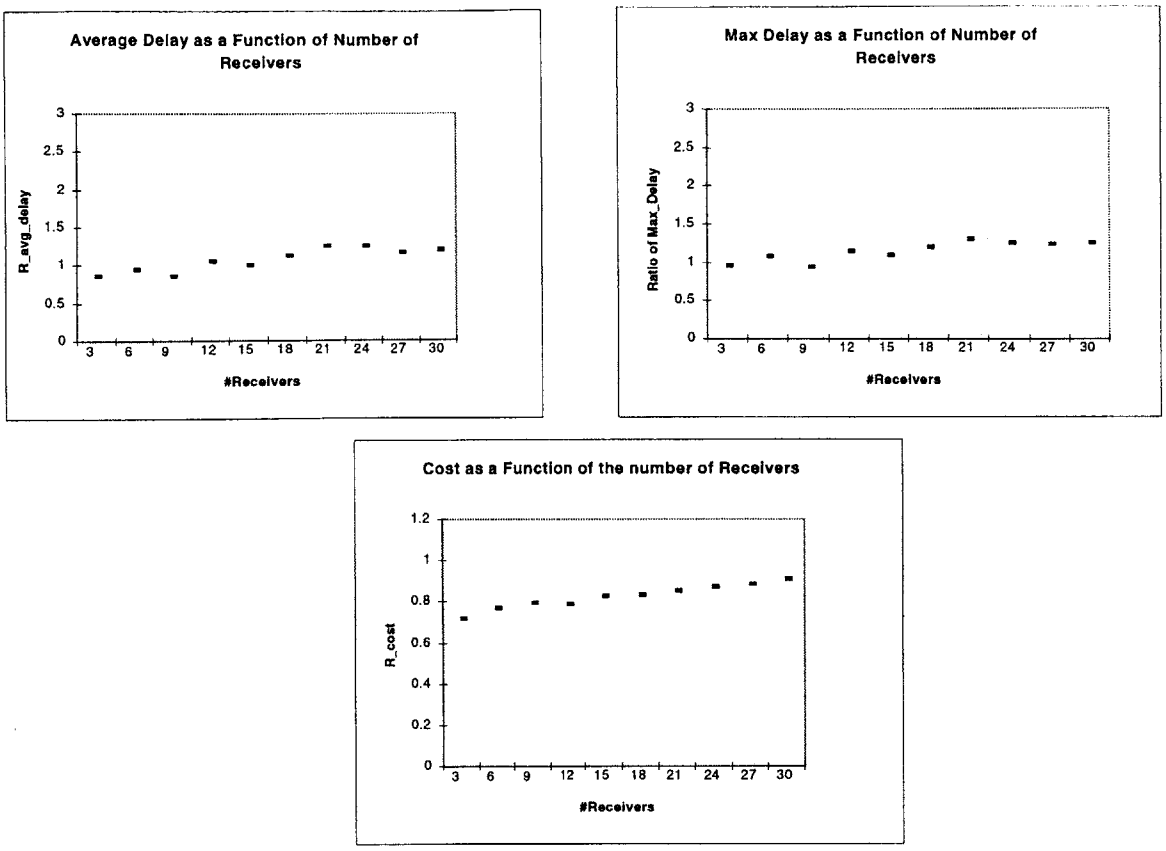


Figure 6-7: The ratio of average delay, maximum delay, and cost in ACBT to their counterparts in CBT as functions of the number of receivers.

Delay and Cost as Functions of the Edge Degree

The results in Figure 6-8 represent the ratio of delay and cost in ACBT to their counterpart in CBT for different edge degrees. The data show slight change in the cost ratio, which we do not think to be meaningful. On the other hand delay might increase. We do not think that the increase in delay is directly attributed to the increase in nodes average degree. We rather attribute the increase to the way we choose the CBT's RP as the node with maximum number of edges in the domain. As the number of edges grows bigger the variance also becomes larger such that we can find RPs with edge degree much higher than the average. Though this improves CBT's performance it is not a likely situation in real networks.

Degree	R_avg_delay	R_Max_delay	R_costs
3	0.81	0.92	0.81
4	0.92	1.14	0.77
5	1.05	1.2	0.81
6	0.94	1.11	0.82

Table 6.2: The ratio of average delay, maximum delay, and cost in ACBT to their counterparts in CBT as functions of the average node degree.

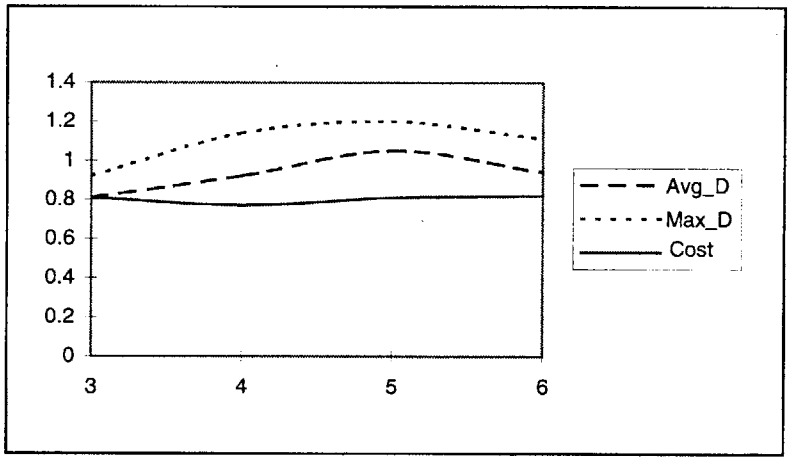


Figure 6-8: The ratio of average delay, maximum delay, and cost in ACBT to their counterparts in CBT as functions of the average node degree.

Delay and Cost in a Hierarchical Topology

Figure 6-9 shows the ratios of delay and cost in ACBT to those in CBT in a hierarchical topology. Comparing Figure 6-9 to Figure 6-5, we conclude that ACBT performs almost twice as well in a flat domain as in a hierarchical one. This conclusion should not be surprising since in a hierarchical domain the potential tree shape is not flexible enough and many nodes have only one way to be attached to the tree.

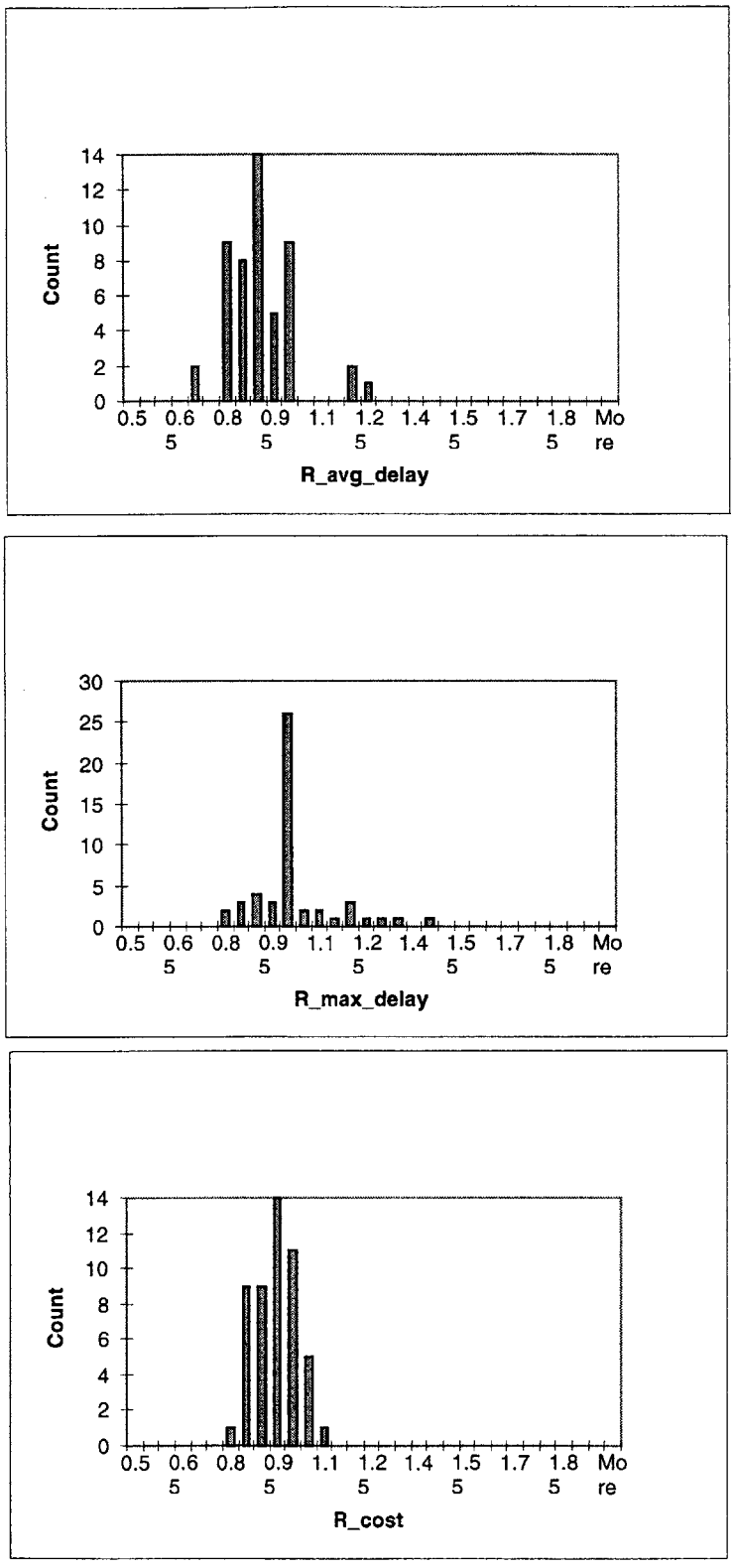


Figure 6-9: The ratio of average delay, maximum delay, and cost in ACBT to their counterparts in CBT in a hierarchical topology.

Traffic Concentration

This simulation is performed on a 50-node graph generated by use of the Doar -Leslie method. 76 groups each with 7 senders and 7 receivers are placed in the domain. The total number of flows that traverse a unidirectional link is counted. 10 simulations were run. The results showed enough similarity that we did not feel the need for more trials.

Table 6-3 shows that by using ACBT we decreased the maximum link load by more than 58%. Figure 6-10, 6-11 give the traffic distribution for entry 3 and 9 in Table 6-3 respectively (the minimum and maximum ratios). It reveals that ACBT has a more satisfactory traffic distibution than CBT.

ACBT	CBT	Ratio
179	428	0.418224
153	430	0.355814
153	432	0.354167
186	446	0.41704
193	446	0.432735
193	444	0.434685
186	425	0.437647
193	427	0.451991
193	432	0.446759

Table 6.3: The ratio of the maximum link load in ACBT to that in CBT.

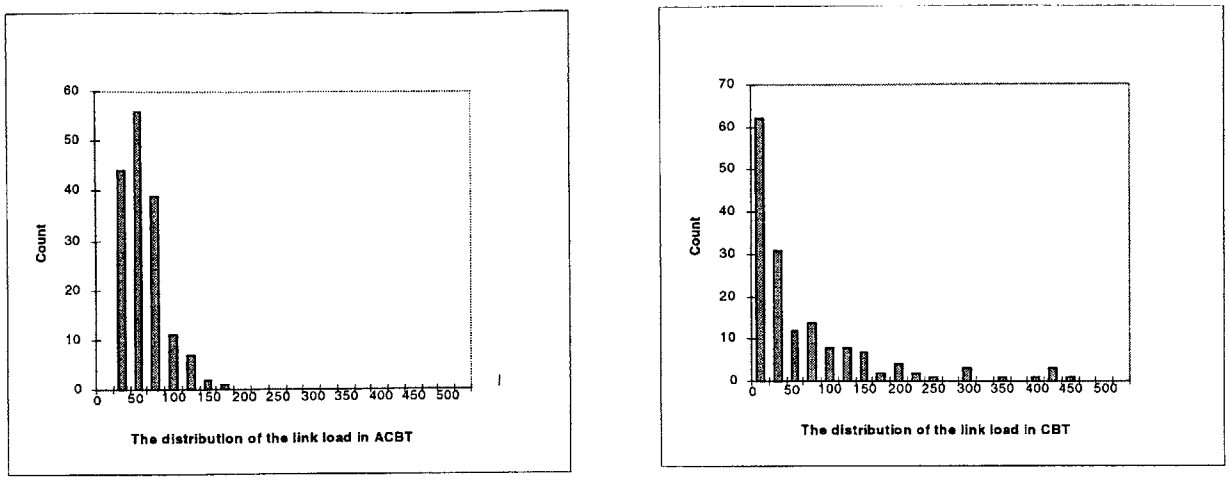


Figure 6-10: The distribution of the link load in ACBT, and the distribution of the link load in CBT for the same set of groups: entry 3 in Table 6-3.

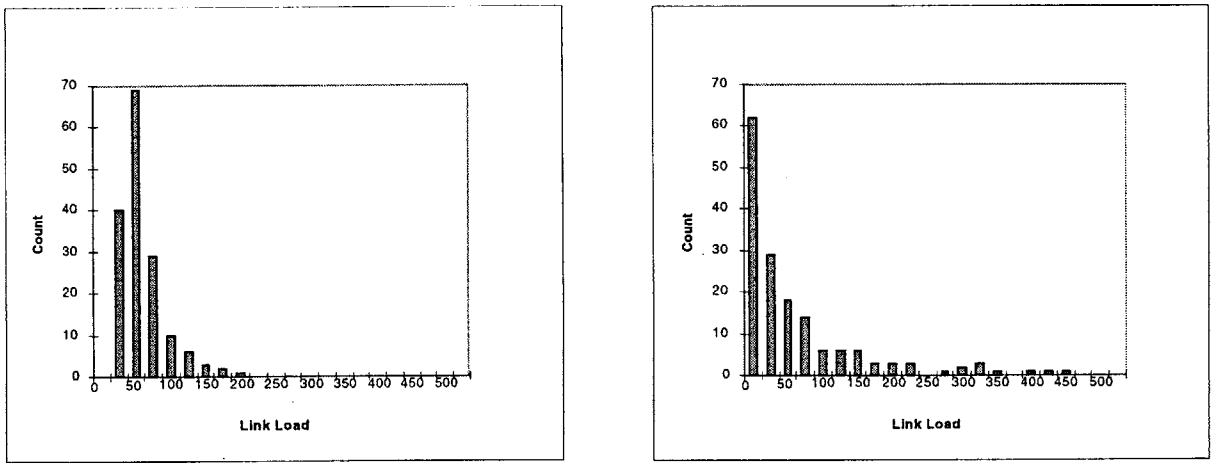


Figure 6-11: The distribution of the link load in ACBT, and the distribution of the link load in CBT for the same set of groups: entry 9 in Table 6-3.

6.4.5 Simulation Conclusions

To drive conclusions from the above data we should decide on how representative they are. The first thing to notice is that anycast in real life is done in a distributed manner. This means that it is possible for a router to be on-tree yet still a nearby joining member does not know about it. From the experimental data available this doesn't seem likely [35]. The second important point is

that ns only counts the transmission time as opposed to the processing overhead. As such a core router with many edges increases the performance of CBT considerably. However in real life one likes to balance traffic in a domain and not overload one router while the others are unloaded. The last point we like to make here is that our simulation investigated cases where groups have no local topology. However, ACBT is most suitable for groups with localized structure. Thus, the simulations reflect a lower bound on the improved performance obtained by use of ACBT.

To summarize the above data, we would say that in a flat domain and for random groups whose size is less than 20% the domain's size ACBT decreases on average the tree cost by a factor greater than 20%. A better performance is expected for localized groups. ACBT improves the average delay, yet it increases the propagation delay's variance.

Chapter 7

FleSc: A Flexible Scalable Interdomain Multicast Routing Protocol

7.1 Requirements for Interdomain Multicast

An interdomain multicast protocol should have the following characteristics:

Scalability

For a multicast routing protocol to scale well, its overhead in terms of bandwidth and state should stay manageable as the Internet expands and as multicast evolves to a larger number of groups with a variety of bandwidth requirements.

On one hand, the protocol should efficiently deliver the traffic to all interested receivers while minimizing the bandwidth consumed by data packets, the control traffic needed to build and maintain the tree, and the routing and forwarding states stored in the routers.

On the other hand, the protocol should place the workload in areas of the network whose resources increase with increased multicast activity[21]. For example, if domains A and B are both multicast enabled and domain A multicast activity increases, then it is desirable to locate the increase in overhead at A since A is most likely provisioned to handle its multicast activity. An approach that increases B's overhead as well is less likely to scale in an environment as heterogeneous as the Internet.

[21] We define multicast activity at a domain by the tuple (maximum number of groups sharing a router, maximum multicast bandwidth on a link).

Stability

At the interdomain level, it is more important to build a stable tree than to constantly reshape the tree to reduce delay or bandwidth consumption. Reshaping can cause loss of packets for sessions in progress and additional control overhead on the routing system. Furthermore, a stable tree facilitates forwarding-state aggregation and reservation making.

Policy

As the Internet becomes more commercialized, a domain's ability to police its resources becomes more exigent. An interdomain multicast protocol should allow each domain to assign resources to multicast traffic according to its own policy, it should not assume that the multicast topology is necessarily congruent with the unicast topology, and it should minimize third party dependencies.

Quality of Service

Quality of service is still an unresolved topic in the unicast context. It is even less clear how it should be handled in a multicast context. However, a QoS friendly architecture should allow the end systems to build alternative branches when the default ones can not accommodate the requested service or cross an untrusted region. It should integrate these branches in the tree without causing loops or starvation.

7.2 Background

7.2.1 The Border Gateway Multicast protocol

Developed by the IETF community, the Border Gateway Multicast Protocol (BGMP) is a promising solution for the interdomain multicast problem [33]. BGMP builds a bi-directional shared tree of domains rooted at the domain owner[22] of the multicast address. To establish such a tree, BGMP relies on two other protocols; MASC, which allocates addresses hierarchically to domains; and Border Gateway Protocol [41], which advertises this allocation and makes it possible to route directly to a multicast address.

Similarly to BGP, BGMP is run by the border routers of a domain, and uses TCP as the underlying communication protocol. Represented by the exit border router towards a group's root domain, each domain functions as one on-tree node that receives joins for that group from downstream domains and propagates them towards the root domain. The interdomain branch becomes fully established when the join hits either an on-tree domain or the root domain.

7.2.1.1 Limitations of BGMP

The BGMP architecture provides a solid framework for the Interdomain multicast. It acknowledges the need for providing multicast policy and addresses both scalability and stability issues. However, BGMP exhibits some limitations.

- **Non-hierarchical topology leads to suboptimal tree layout**

To scale, BGMP relies on MASC to allocate addresses hierarchically. This allocation is advertised through BGP to generate multicast routes along which join messages are forwarded. Since the Internet topology is not really hierarchical this architecture creates a tradeoff between a larger number of multicast routes and a better tree layout. To make this clearer, take the example in Figure 7-1. Domain D, which is multihomed to B and C, is the root domain for group G. Domain E is interested in joining G. If D advertises G only through its parent C, then the branch from E will be E-B-A-C-D. If D advertises G to B then the branch from E will be E-B-D. However, since group G is not aggregatable within B's multicast address space, advertising G to B increases the size of the multicast routing tables and the bandwidth consumed by BGP to advertise and maintain the multicast prefixes.

[22] The domain whose allocated multicast prefix covers this address.

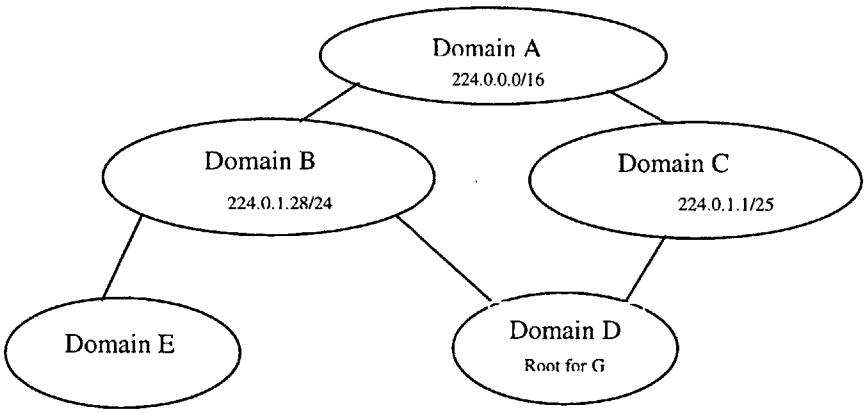


Figure 7-1: The impact of multihoming on BGMP.

We are not certain how significantly this non-hierarchical topology will contribute to poor scaling or inefficient multicast routes in the long term. However, it is interesting to note that currently more than 25 percent of the networks in the Internet are multihomed and that the rate of increase in multihoming is at best linear [42]. We have no explicit data on the level at which multihoming occurs[23], but the data presented in [42] allow us to carry the following reasoning: Four types of domains exist in the Internet. C1 domains which constitute 0.9% of the total number of domains and which are national or international backbones. C2 domains, which constitute 3.1% of the total number of domains and which are regional US providers or European national providers. C3 which constitute 9% of the number of domains and are metropolitan area providers. C4 which constitute 87% of the number of domains and are multi-campus corporate or academic networks. On the other hand, nearly 25% of domains have edge degree greater or equal to 3. Thus, at least 12% of the Internet domains are C4 multihomed to at least 3 providers. Furthermore, since 50% of C4's links are to C1 domains we can expect many of these C4 multihomed domains to be multihomed to C1.

In fact, multihoming is only an example where the hierarchical-topology assumption fails. Currently, there are evidences that the Internet is significantly non-hierarchical (25% of all the

[23] At what level the multihomed domain's prefix is reintegrated in the antecedent prefix.

links in the Internet are between C1 and C4 [42].) which makes any architecture that relies on hierarchy to scale questionable.

- **Routers are overloaded with a large number of multicast routes that they might never use.**

Even if there is no single active multicast group in the Internet, the routing system still has to advertise the multicast prefixes to the entire Internet and maintain the multicast routes. This is undesirable given that, in practice, the number of active[24] interdomain multicast groups at a domain is much smaller than the total number of multicast routes.

- **BGMP is prone to transient loops**

Multicast loops are very destructive since a packet that enters a loop generates other packets that loop with it and generate more packets. To be loop-free, BGMP relies on the assumption that at any point in time there is consistency in the routers' view of which BR is the exit point towards the root domain of each multicast group. However, the fact that the protocol uses BGP to distribute the multicast routes creates periods of inconsistency after which the system converges to a consistent view of the network. To see how this could create transient loops consider the following example:

[24]An active multicast group is one that is handling traffic. An inactive multicast group is a multicast address that currently has no associated distribution tree.

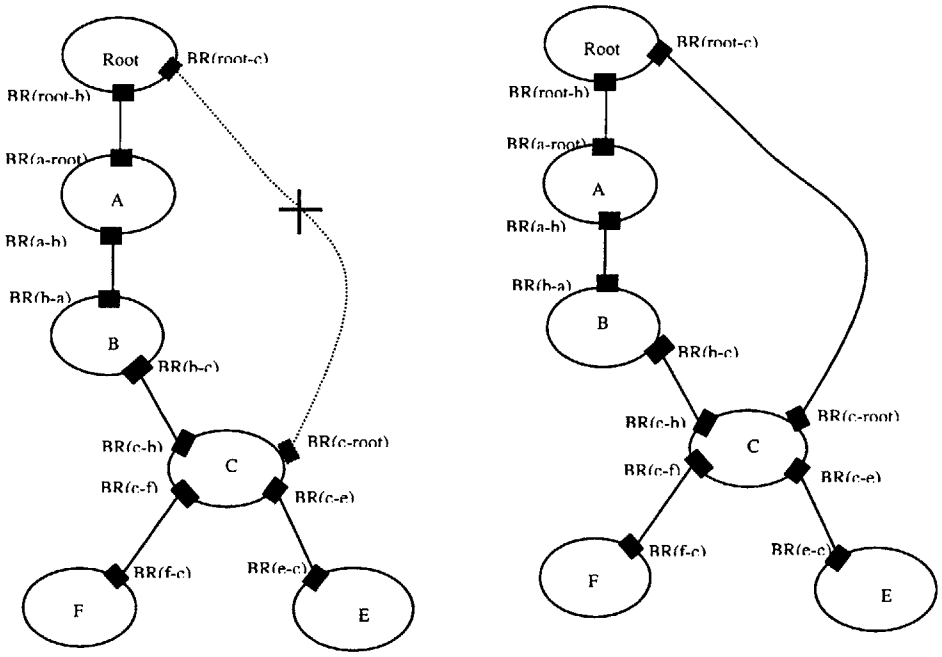


Figure 7-2: Transient loops in BGMP.

Figure 7-2: shows the interdomain multicast tree when link(Root-C) was down and after it becomes functional. While the link is down, domains E and F both join the multicast tree, building the branches E-C-B-A-Root and F-C-B-A-Root respectively. When the link comes up again BR(c-root) is the first to notice the change and starts to propagate the new route to the other BRs in domain C. However, both updates sent to BR(c-b) and BR(c-f) are dropped or delayed by congestion while the update sent to BR(c-e) arrives. In this case, domain E generates a periodic join that travels along E-C-Root. Domain F also generates a periodic join but since BR(c-f) didn't receive the advertisement of the route C-Root, it propagates the join toward BR(c-b). As a result, Domain C gets connected to the bi-directional tree at two different BRs which both inject multicast traffic into the domain[25].

[25] How the loop is treated internally to domain C is dependent on the intradomain multicast protocol C uses. If C is a CBT domain that uses the Bootstrap protocol or a priori configured core, then it is quite possible that it will close the loop.

- **The soft state used to maintain the tree makes it hard to establish any reservations or to choose an alternative route[26].**

BGMP, like PIM-SM, maintains the tree by requiring receiver domains to send periodic join messages towards the root domain. The periodic joins refresh the forwarding state and update the tree's shape according to the changes in the multicast routes. For example, assume that a domain D is a receiver domain that has joined the multicast tree. After D has joined, some route updates cause the best route from D to the root domain to change. The next time D sends the periodic join the message will be delivered according to the new route from D to the root domain, building a new branch and pruning the old one. The advantage of such a process is presumably building a better branch, however reshaping the tree might not be the wisest thing to do when the old branch is still functional. Reshaping causes communication disruption and loss of many of the packets in the long interdomain pipe. Besides communication disruption, the soft state makes the architecture less amenable to reservation establishment since any route change might reshape the tree and cause the reservations to be left behind.

7.2.2 YAM

Proposed by Crowcroft and Calberg, YAM is an interdomain multicast protocol that builds shared trees using one-to-many joining mechanism. The architecture is split into two tiers. First, a leaf router that receives a join contacts the DNS to map the multicast group to its domain's egress node. Upon receiving the mapping, the router joins the egress node establishing the intradomain portion of the branch. To establish the interdomain portion of the branch, the egress node multicasts an interdomain spanning join message. An off-tree node that receives the spanning join broadcasts the message out to its other interfaces via RPF (Reverse Path Forwarding.) When an on-tree node receives the spanning join, it joins back towards the originating egress node. This join installs transient state along the path. Eventually, the originating egress node learns a set of possible routes to the tree. The node selects the best route and sends back an acknowledgement that confirms the transient states along the chosen path.

[26] We use the term pinned tree for a tree that does not change shape unless one of the on-tree links fails (i.e., CBT.) We use the term soft tree for one that is reshaped whenever the underlying unicast routing protocol changes routes (i.e., PIM-SM.).

7.2.2.1 Limitations of YAM

According to our knowledge, YAM is the first protocol to introduce the one-to-many join as a method to route to a multicast tree. This mechanism generates a variety of on-demand multicast routes, which can accommodate a diversity of QoS and policy requirements. However, the protocol ignores the need for multicast policy and relies solely on on-demand generated routes, which may introduce indefinitely long delay and a large amount of control traffic.

On the other hand, it is not clear whether the protocol builds unidirectional or bi-directional trees. If the protocol builds unidirectional trees, then it unnecessarily increases third party dependencies and exacerbates the distant core problem. However, if the protocol builds bi-directional trees then it runs the risk of causing starvation and race conditions[27].

Figure 7-3 illustrates a possible problematic situation. The leaf routers in domain A join to the egress node. However, since the group-egress-node mapping does not take into consideration the relative position of the egress node to the tree, it is quite possible that all of the egress node's routes to the tree cross its intradomain descendants. In this case, the descendants would either discard the spanning join causing starvation, or they join back to the originating egress node causing permanent loops.

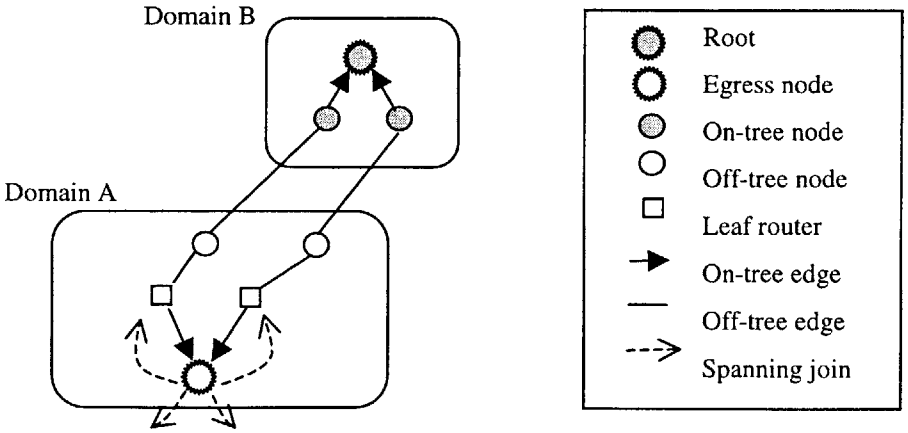


Figure 7-3: Starvation in YAM: the egress node can't join the tree because all the possible routes cross its descendants.

[27] In fact, a tree establishment process that builds bi-directional trees share some similarity with CBT with multiple cores – an approach that has been proven to cause starvation race condition and loops [39].

7.3 Our Approach

7.3.1 Introduction

The multicast routing problem can be divided to two complementary tasks: locating the multicast tree and establishing the multicast branch. Locating the multicast tree is the control process by which the network decides how to forward the join message such that it will eventually hit an on-tree router. It is the process that generates the multicast route, and determines the eventual multicast branch. Traditionally, locating a multicast tree was done by associating it with a routable unicast address (the core address). This means that there must be a mechanism to map a multicast group to its core's address, and there is only one possible branch to attach the receiver to the tree, which is the branch carried on the spanning tree routed at the core router. Different protocols propose different mapping mechanisms. The Simple multicast protocol proposes out-of-band mechanisms such as DNS sdr or the web, PIM-SM uses the Bootstrap Protocol, and BGMP relies on BGMP to provide multicast routes.

On the other hand, locating the multicast tree by itself is not enough to deliver multicast traffic to the interested receivers. Path selection must be accompanied by a process to establish and maintain the necessary forwarding state inside the routers. This is what we call the branch establishment process[28]. To scale the branch establishment process, many argue that it is necessary to aggregate the forwarding states for different multicast groups that share the same path. In this thesis, we do not discuss such aggregation, though we believe our architecture does not impose any constraints on future solutions for the problem.

7.3.2 Assumptions & Observations

Our design relies on the following assumptions, justified by the scarce data available [35] and common sense.

[28] In practice the two processes could be carried at the same time, however, conceptually they are different. They constitute the multicast analogy for routing & forwarding.

1. Many interdomain multicast groups will exhibit locality. For example, a TV broadcast in Paris, although it might have an Internet-wide scope (global address), practically will be centered in France with some receivers in other countries and in other continents. This locality springs from an observable locality in interests, languages, and cultures. Even more important, the difference in time zones across the world drives groups to be local at any point in time. For example, the STS-63 space shuttle mission, one of the most popular MBone sessions, attracted receivers from other countries (around 20% of all the receivers.) However, most of those international receivers joined at noon hours in their time zones. At that time, the American receivers showed low activity [35].
2. We model the Internet as a set of connected domains. We differentiate between leaf domains where we expect most of the senders and receivers to be, and transit domains, which are mostly service providers. We expect leaf domains to run the interdomain multicast protocol at the border routers while running their preferred intradomain multicast protocol internally. Our interdomain multicast protocol is built of two modules: the tree location module and the branch establishment module. In transit domains, we expect the border routers to run both modules and the internal routers to run at least the branch establishment module. In case the internal routers in a transit domain do not understand the tree establishment module, it is still possible to integrate the domain in the interdomain multicast tree at the expense of the domain simulating a single router[29].
3. There will be a large amount of overlap in multicast and unicast topologies. Although, the current multicast topology (the MBone) is not congruent with the unicast topology, we believe that the more pervasive multicast becomes, the more the topology will approach the unicast topology. This is because administrating one virtual topology is always easier than administrating two of them[30].
4. Similarly to BGMP, we assume the existence of a multicast allocation protocol that assigns addresses hierarchically.
5. We assume that multicast reachability information is advertised through MBGP [41]. However, we assume that the routes are advertised based on the unicast prefix rather than the multicast prefix. We also assume that those routes are bi-directional (necessary to build a bi-

[29] A method for doing this can be found in [18].

directional tree) and are group independent[31] (It is worth noting that group independent routes do not allow exercising policies according to the multicast address. They allow policies that are based on the path the route crosses. However, the only information embedded in a multicast address, at this stage, is its owner domain[32]. Later, the address would represent a group with certain characteristics but the address does not tell us anything about the type of groups it might be assigned to. Thus, using group independent routes does not limit our policy space since no information existed in the address that is suitable for additional policy rules.)

7.3.3 Design decisions Based on the Requirements

1. Bi-directional Distribution Trees

The need for bi-directional distribution trees springs from both the scalabilty and the policy requirements. Many previous simulations revealed that bi-directional trees have potentially smaller path length than unidirectional trees [38, 23]. Moreover, bi-directional trees perform much better when senders are receivers, a common situation at the interdomain level since leaf nodes are domains. Even more important, bi-directional trees reduce third party dependencies. Members in two domains can communicate together without having their communication be dependent on the quality of their connectivity to the root domain. Their communication depends only on the quality of connectivity to domains lying on the shared path between the communicators.

2. Pinned Trees

Tree maintenance in soft state protocols, like PIM-SM, relies on the periodic generation of the join messages, which reassure on-tree routers of the presence of interested downstream receivers. On the other hand, pinned state protocols, like CBT, maintain the tree by generating one-hop keepalive messages that ensure every parent of the existence of its children, and vice versa.

---

[30] This does not mean that the two topologies will be exactly the same.
[31] A group independent multicast route is a route that can be used to forward traffic to any multicast group.
[32] We are talking about interdomain multicast; thus, our discussion does not involve scoped addresses.

The advantage of a soft state protocol is keeping the tree's shape consistent with the latest unicast routing updates. However, we think that, at the interdomain level, this constitute more of a burden than a benefit. First, they work against the stability requirement, since tree reshaping causes additional control traffic and a communication disruption. In practice, the improvement in the tree shape may not be worth the overhead. Second, soft state makes it harder to establish reservations along the tree. For example, a receiver might try to reserve bandwidth at a bottleneck router, or even build an alternative branch to avoid the bottleneck router. However, the tree changes shape and the bottleneck moves to a different point, and the reservation are either left behind (on a pruned branch) or become useless (since they do not affect the receiver's share at the new bottleneck.)

On the other hand, pinned state protocols conserve the same tree as long as all on-tree links are functional. When an on-tree link goes down, the downstream subtree is flushed. Detached receivers rejoin the tree on their own.

3. Interdomain Multicast Routes

The current interdomain unicast reachability information provided via BGP enables a domain A to convey to a domain B information such as "you can use this route to send traffic to the destination". However, such reachability information is not useful for establishing either unidirectional or bi-directional multicast trees. For unidirectional trees, we need information such as "you can use this route to receive traffic from group G (or source S)". For building bi-directional trees we need both types of reachability information. Moreover, these routes convey the interdomain unicast policy, which might be different from the interdomain multicast policy. Thus, we need explicit multicast routes along which we can send join messages.

Our protocol establishes bi-directional pinned distribution trees along multicast routes that are provided by an underlying protocol (BGP, for example.) It assumes that multicast routes are bi-directional and, by default, group independent. In other words, it assumes that multicast routes are based on the unicast prefixes[33]. This decision enables the aggregation of the multicast and

[33] This also means that the routes are based on the anycast prefixes, since anycast addresses are obtained by concatenating the anycast indicator with unicast prefix and the group ID.

unicast advertisement when the two topologies are congruent (the route is tagged as unicast and multicast enabled.)

4. Customized Routes According to the End Domain's Interests

Our architecture relies on a relatively old concept introduced by Clark, which says that to scale a heterogeneous unlimited size nonhierarchical network one should distribute routing information according to interests rather than according to a virtual assumed hierarchy. We think that this concept is particularly suitable for locating a multicast group (tree) at the interdomain level. We state the following reasons for this suitability:

1. The number of multicast groups a domain will simultaneously join is smaller than the total size of a multicast routing table (all multicast routes.) The number of active multicast groups in a domain is limited by the domain's capacity and by the intermediate routers' ability to handle all the necessary forwarding state.[34]
2. Even more important, the overhead a domain carries is proportional to the number of multicast groups it joins, and is not affected by other domains' multicast activity. This enables a smooth transition from the current situation where multicast is hardly used to the future Internet when multicast becomes very popular. Moreover, the domain's ability to join additional multicast groups is restricted by its own capacity. It is not affected by external parameters such as the number of multihomed domains in the Internet.
3. For a protocol to distribute routing information according to the end domains' interests and to be able to forward any packet to its destination it has to decouple routing and forwarding[35]. This can be accomplished by having end systems either encapsulate their packets or inject the necessary state in the intermediate routers. In the case of unicast, encapsulation induces a per packet overhead, which makes it undesirable for long sessions. In contrast, injecting the route in the intermediate routers is done by the first packet, which makes it reasonable for transferring a lot of traffic. However, if the connection transfers few packets the state and latency overhead becomes cumbersome. In the case of multicast, since the routing protocol must build a tree, it has to experience some latency and it has to inject state in intermediate

[34] The fact that receivers are usually in stub domains means that most of the multicast trees that a domain joins will share some intermediate routers.

[35] Since intermediate domains might not be interested in the same routes that downstream domains are interested in.

routers. This means that multicast is intrinsically unfavorable to very short sessions. However, this tree establishment process, by injecting the forwarding state along the path, provides a natural mechanism to decouple routing from forwarding. Injecting the multicast route by the end domains does not increase the state or latency overhead since the branch will be established whether routes were originally distributed to everybody (BGMP approach) or distributed only to people interested in this group (our approach.)[36]

Therefore, To achieve high scalability, FleSc allows each end domain to customize its multicast routes according to its interests. As such, transit domains maintain hierarchical aggregatable multicast routes. On the other hand, each stub domain expands the hierarchical routes with a set of on-demand generated multicast routes to groups that it is interested in joining.

7.3.4 The Architecture

Assuming that multicast addresses are hierarchically allocated to domains, our protocol builds bi-directional shared trees rooted at the domain whose multicast address space includes the group's address. To pull data down, leaf domains conduct a localized search for a nearby on-tree domain. If the search succeed the branch is created according to the generated route. If the search fails the join is forwarded according to the hierarchical routes.

Locating the multicast tree

To locate a multicast tree we rely on the following principles:

1. Generate cheap hierarchical routes for inactive multicast groups

Similarly to BGMP, we think that multicast addresses should be allocated hierarchically and that the allocation should be advertised globally. However, instead of advertising group routes we advertise multicast enabled routes, which are carried in the unicast advertisements by tagging a route as multicast enabled. When a route is exclusively for multicast traffic a separate

[36] In the coming sections we will describe a method by which down stream domains learn about routes to groups they

advertisement is sent. We advertise the binding of multicast addresses separately.[37] As such, a multicast route is a path of adjacent ASs over which a multicast branch can be established to distribute traffic for any multicast group. Furthermore, we only advertise hierarchical multicast routes.

Why are these routes cheap? First, most of the routes are advertised for free in the unicast advertisement by tagging them as multicast enabled. The binding, which we need to advertise, is much more stable than the routes themselves, and is rarely updated.[38] Last, the hierarchical allocation of addresses helps aggregate the entries in the multicast forwarding tables.[39]

2. Generate high quality multicast routes on demand.

Establishing the multicast branch over the aforementioned cheap routes incurs two deficiencies. First the routes are hierarchical, as such they ignore many shortcuts that potentially could make the branch much shorter (see the case of the multihomed domain in Figure 8-1). Second, the cheap routes tell us only how to locate the root of a multicast tree. However, in many cases a domain will find it more convenient to build a branch to a non-root on-tree node, either because the resultant branch is shorter, or because certain QoS requirements can't be satisfied along the route to the root domain, or because of some policy considerations.

To assess the value of on demand route-learning it is important to know how far we should go to find a route to a multicast group, and how big the space of useful routes is (how many useful branches could be built between this domain and some on-tree node.) In the unicast context, the study in [42] shows that the diameter of the Internet equals 10 domains, and that the diameter is staying constant despite the Internet's huge growth. This implies that no matter what the destination we want to route to is, in today's worst case, we need to send probe messages for a distance shorter than 10 domains. Moreover, the study in [42] shows that there is a high interdomain-path redundancy and that on average there are nearly 22 useful[40] paths to a domain

---

are interested in joining.

[37] The binding indicates which multicast address belongs to which domain.

[38] Note that if you allocate the multicast prefix to indicate the unicast prefix of the domain, then you do not even need to advertise the binding.

[39] Note that the hierarchical allocation of addresses is less important in our scheme than it is in BGMP. Thus, if it turns out that it is very complicated in practice, the system still can manage without it.

[40] A useful path is one whose length differs from the shortest domain level path by at most 1.

from a fixed backbone provider.[41] Probably some of these 22 routes do not comply with the desired policy. However, most of them are deserted because of the coupling of the routing and forwarding in the Internet. Given the above mentioned information and the locality of multicast groups, we expect a route search for a multicast group to succeed with a high probability, to take little, time and to find a large variety of useful routes.

3. Delegate route generation to the end domains

Above we argued that it makes sense to distribute multicast reachability information according to the end systems interests and that by probing the network an end system can with high probability and in a little time locate many nearby on-tree nodes. Here we give more information on how a domain conducts such a search[42].

For a domain to locate a multicast tree, it broadcasts a search message that hops domain by domain collecting information about the possible paths. The originating domain includes in the search message a counter set to the maximum number of domains the message will cross (generally limited to 3 or 4 since most stub domains are closer than 4 domain-hops to the backbone [42].) The domain indicates in the search message whether the join should collect information at the domain level or at the router level (for example, how many router-hops the path is.) The domain also indicates whether the message should collect information about the path performance. The search message is propagated to all adjacent domains using a mechanism similar to the one BGP uses to send its route advertisements (over TCP connections and complying with the same rules BGP messages comply with when traveling between external and internal peers.) Each domain that receives such a message has to answer the question whether it is on the multicast tree or not. The answer should be available to the domain's BRs through Domain Wide Reports [37]. If the answer is "Yes" or the domain receiving the message has started a search for the same group, then the message is stopped and the information collected via the search is unicasted back to the originating domain.[43] If the answer is "No" then the message

[41] Furthermore, this number has more than doubled from 1994 to 1995.

[42] Note that FleSc does not rely solely on on-demand route generation. It falls back to the hierarchical routes whenever it fails in finding local on-demand routes.

[43] To make the argument clearer we didn't go through describing the detail of the process, later we provide more detail. However, the thing to keep in mind is that this is similar to the border routers learning an exterior routes for certain destination and deciding on one of them and then advertising the route adopted to adjacent domains.

is propagated to adjacent domains after decrementing the counter. When the counter reaches zero the search message is discarded. When propagating the search message each domain applies its policy on the collected route contained in the message. The message is forwarded further only if the route it contains complies with the forwarding domain's policy. Furthermore, the message is forwarded only to adjacent domains from which the domain is willing to receive multicast traffic (and obviously, to which the domain is willing to forward multicast traffic, given that the tree is bi-directional.) Thus, each domain along the path executes its policy by accepting the join and choosing to which adjacent domains to propagate it.

4. Use pre-fetching

In computer caching systems, the CPU predicts that an access to a certain memory location will be followed by an access to nearby locations, and accordingly the CPU prefetches the nearby memory blocks. We think that multicast groups create a similar situation. Since groups are ephemeral and localized we can predict that when a domain issues a search message to locate a multicast tree, there is an increased probability that in the short following interval nearby domains will issue search messages for the same multicast group. In other words, after a multicast session is advertised, interested users start joining. Most of the joins will arrive in the short interval[44] before the session starts. Given the assumption that most multicast groups are localized, it makes sense for an on-tree domain that is hit by a search message to advertise to neighboring domains the multicast group.[45] Domains that receive the advertisement treat it similarly to other reachability information (the BGP way.) The only difference is that they do not propagate the advertisement outside the neighborhood of the advertiser (the advertisement includes a counter that is decremented every domain-hop and is discarded when the counter reaches zero.) Also, on-tree domains and domains that have a contradictory policy do not propagate the advertisement.

The route the advertisement establishes has an associated timer whose expiration causes the route to be discarded. Moreover, if the advertising domain leaves the multicast group before the timer expires, it should withdraw the route it has advertised.

[44] It could be hours. However, it still short compared the lifetime of the hierarchical multicast routes.

Establishing the multicast branch

1. Send a directed join toward the discovered on-tree domain

When a domain originates a search message it sets a timer whose expiration causes the domain to inspect the set of candidate routes it has learned. The domain picks the best candidate according to its own policy. All else being equal, it is always recommended to chose the route in the direction to the root domain of this multicast group.

To establish the branch the domain sends a directed join message that includes the learned route as a vector of m-nodes. In the simplest case the m-nodes will be domains each represented by the entrance BR and the exit BR that the search message has crossed. In a more sophisticated case, each domain in the vector path is described by the set of routers that constitute the intradomain portion of the branch.[46] Note that the path vector carries information at different levels of abstraction. Moreover, it carries enough information such that each node in the vector when it receives the join, knows exactly how to build the portion of the branch between itself and the next node.

In case the domain failed in generating a route[47] it sends the join towards the root domain. The path vector in this case includes only one m-node, which is the root domain.

2. Handling joins by routers

Two things should be considered carefully while handling a join. First, although each domain chooses the branch it wants to build, the eventual branch might be a sub-branch of the intended one. In other words, to prevent loops the join should be stopped by the first on-tree router it hits, and the resultant branch is the portion of the path vector connecting the receiver with this on tree router. If the receiver domain wants any particular node in the path vector to necessarily be on the branch[48], it must tag that node as mandatory in the interdomain join. In the case the resultant branch does not include some mandatory nodes, the router at which the establishment of the branch has stopped flushes the sub-branch and informs the originating BR that its request can not

[45] The domain uses a timer to prevent advertising the same group multiple times in a short interval.
[46] This refers to a more detailed route (similar to the map with different levels of abstraction introduced by Nimrod)
[47] Or if the domain does not want to generate a route.
[48] Probably the node is necessary for QoS reasons.

be satisfied. In this case, the originating BR can use an alternative learned route or the default one[49].

Second, the fact that joins are directed toward different on-tree routers forces the system to take precautions to prevent race conditions. More precisely, in a system that directs all joins for a particular group toward the same core, a join either hits an on-tree router or loops and eventually times out. However, when joins are directed toward different destinations a race condition is possible. Figure 7-4 illustrates this situation. Router1 receives a directed join from DR1. It sets up transient state and forwards the join along the direction indicated in the path vector. Router2 receives a join from DR2 and sets up transient state and forwards the join toward its destination. The join coming from Router2 crosses Router1, which stops it because it has already a transient state for that multicast group. Furthermore, the join forwarded by Router1 crosses Router2, which stops it because it has a transient state for the multicast group. Both transient branches will time out eventually, and neither receiver will get on the tree. If DR2 and DR1 simultaneously reattempt to join the tree then the same situation reoccurs.

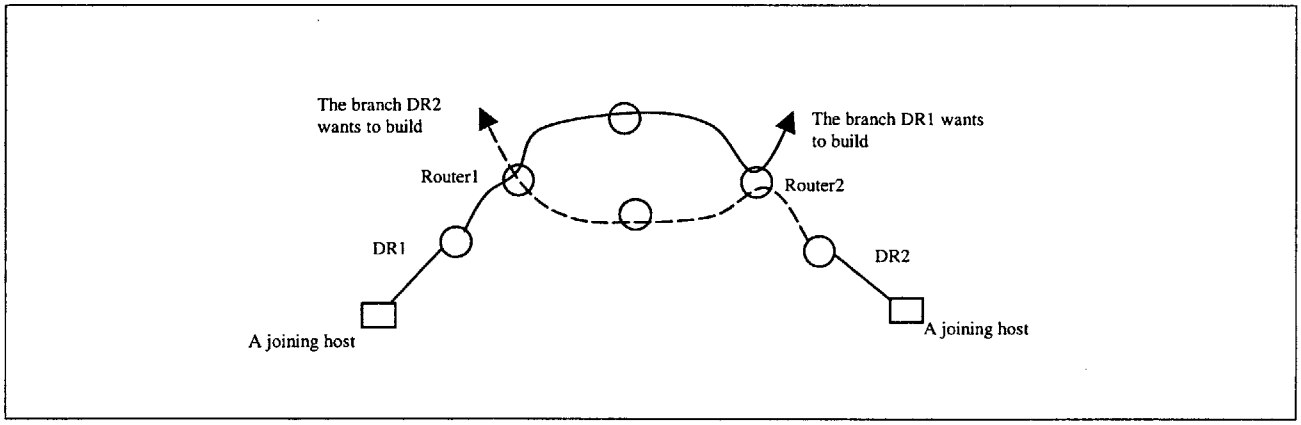


Figure 7-4: Illustration of a race condition caused by a naïve usage of a non-minimum-spanning tree

[49] It might seem that the on-tree router should try to establish the mandatory branch by propagating the join upstream waiting for the ack and then leaving its first parent. First, this works against the stability requirement and might cause communication disruption. Second, this is too cumbersome since in the absence of additional information the router might join to a downstream node causing a permanent loop. Our decision aligns with the principle of fate sharing, which should be expected when building a shared tree.

To solve the above problem, a router that has transient state for a multicast group and that receives a join for the same group that is not directed to its parent, has to break ties[50]. If the new join wins the tie-breaking then the router quits its parent and forwards the new join along toward its destination. The router, however, keeps the old pending state since the join-ack, which will be received from the new parent, confirms both pending states.

7.3.5 Putting It Together

- Joining a multicast tree

A receiver expresses its interest in joining a multicast group to its first hop router (DR). If the DR decides that the multicast group is external it unicasts the join to the border router responsible for this multicast address. This BR could be a configured one or the exit BR toward this multicast group. If the BR is not already searching for the group, it sets a timer and contacts the other BRs in the domain asking them to start the search process. Each BR unicasts the sets of routes it finds to the BR which started the search. When the timer fires the BR inspects the routes it found and chooses the best one according to its policy and possibly QoS requirements. The BR responsible for the multicast group asks the exit BR along the chosen path to generate a join message that includes the chosen vector path. The responsible BR makes sure that all previous joins are rerouted, and all later joins are delivered to the exit BR along the chosen path. How to do this is domain dependent. For example, if the domain is a CBT domain or a PIM-SM domain then the exit BR should advertise itself as the core router for the multicast group. For a DVMRP domain it is enough for the BR to pull down the data and act as a proxy-source for it (as long as the BR passes the reverse path check. Otherwise, the data should be encapsulated and injected into the domain via the exit BR toward the source.)

The exterior portion of the tree is established by the interdomain join message. The join establishes a transient state in each router it crosses. This state includes the previous hop router, the next hop router, the final destination of the join, and the originator. The join is stopped by any on-tree router it crosses, and a join-ack is sent back along the newly established branch. When a

[50] possibly by comparing the IP addresses of the joins' destinations

router that has a transient state for the multicast group receives a join message with a final destination whose ID (AS-number/IP-address) is greater than the final destination in the transient state, it quits its parent, forwards the new join to the next hop towards its destination, and updates its transient state to indicate the new parent. The router keeps the transient state it had since the join-ack confirms all transient states for the same multicast group.

If the search fails and the timer expires before finding a route, the join is sent toward the root domain for the multicast group according to the cheap multicast routes, in which case the vector path has only the root domain's ID (AS-number).

When the join hits an on-tree router a join-ack is generated and sent over the same path the join traveled. When the join-ack reaches the originating BR, the domain can receive the multicast traffic.

If the last node in the join vector path is a domain and not a router, then the BR receiving the join has to join the tree internally according to the intradomain multicast protocol rules.

- Non-member senders sending to a multicast group[51]

Sending to a multicast group is similar to locating a multicast tree. Thus, again we differentiate between internal groups, which have distribution branches in this domain, and external groups, which have no on-tree members in this domain.

If the group is internal then the intradomain multicast protocol should deliver the traffic to one and only one internal on-tree router. Once the packets hit an on-tree router they get forwarded downstream and upstream to all receivers. For example, in a PIM-SM environment the border router that pulled down the traffic to the domain should advertise itself as the local RP for the multicast group.[52]

If the group is an internal group in a transit domain that uses the interdomain multicast protocol only, and if the domain expects local senders for this group,[53] then again one of the on-tree border routers advertises itself as the group's local RP.

---

[51] Member senders generate native multicast packets, which are forwarded along the tree.

[52] Note that this is similar to the method suggested in BGMP specifications.

[53] Note that transit domains in most cases are provider domains and do not have senders or receivers. We think in practice transit domains should consider the group external even if they have an internal branch. Then when an internal sender sends traffic to the group, the status of the group changes to internal and one of the on-tree border routers advertises itself as the local RP for that group.

Packets generated by non-members senders and destined to an external multicast group are forwarded to the exit BR for this group, which forwards them along the cheap multicast routes to the group's domain-root[54]. For most applications, we expect sending to an external multicast group to be uncommon since it means that the group has no receivers in the sender domain.

Note that although the choice of the root domain is less critical in our approach than in BGMP, it is highly recommended for applications that have one sender (e.g. TV broadcast) to locate the root at the sender's domain.

7.4 Discussion & Evaluation

7.4.1 Overhead

- **Increase in the worse case join latency**

In the best case the search process terminates at an adjacent domain returning an acceptable route over which the branch is established. In the worse case the search process fails, causing the join message to be forwarded to the root domain over the cheap multicast route. Therefore, in the worst case the join latency will be extended by the timer interval. Note however, that this is only true for the first receiver that joins the multicast group. Other receivers join to internal on-tree routers and do not encounter any additional delay. We know only of one class of applications for which this latency is unacceptable, which is Distributed-Interactive-Simulation. However, these applications have numerous scalability problems even when run in a local network environment. We doubt that they can scale to the global Internet. We think that the default behavior at the interdomain level should be to trade join latency for scalability and flexibility. However, it is always possible to pre-join as a means of escaping the delay incurred by the search mechanism.

- **Internal routers in transit domains are required to understand the interdomain join semantics**

[54] At the interdomain level, it is less likely that a domain is only a sender. Moreover, in case of applications with one sender, the sender can join the group to enhance its path to the tree.

For a transit domain to run its own intradomain multicast protocol while being part of an interdomain multicast tree it has to interact with the outside world as one entity. It has to have a consistent view of what is the next hop toward the destination of a join message. We know of two approaches to handle this. The first is BGMP's approach, which relies on a dynamic mechanism to define which is the exit border router toward the root domain. We have seen in section 8.2 that this approach can result in transient loops. The second is HIP's approach [18], which relies on a configured router to act as the domain representative for matters concerning a particular multicast group. The drawbacks in this case are the delay necessary to contact the representative router, and the lack of robustness in the system caused by the configuration. The problem of consistency is exaggerated in case the protocol accepts joins for the same group to be directed toward different destinations.

In our architecture we choose to make the internal routers in a transit domain aware of the interdomain join semantics. It is still possible to use an approach similar to that used by HIP to present a transit domain as one node that runs whatever intradomain multicast protocol is desired internally, yet provides a consistent view of its state to the surroundings.

7.4.2 Advantages

Most of the reasoning behind the advantages is stated in the architecture section. Therefore, we only restate the advantages without further discussion.

- **Potentially smaller trees**

The search process enables a domain to join the multicast tree at the closest node building a small branch and reducing the total size of the tree. This reduction in the tree size implies directly a reduction in the bandwidth and state necessary to deliver the multicast traffic. This greedy approach usually decreases the tree cost by 20% to 30% compared to other shared tree approaches [36], however, we expect an even higher reduction due to the localized topology of interdomain multicast groups.

- **Reduction in the amount of information stored in the routers**

In our protocol routers only store the cheap hierarchical multicast routes. The hierarchical structure of the routes makes aggregation possible and reduces the amount of memory necessary to store them. Compared with BGMP, our protocol uses less state overhead in the routers, however, it generates a larger variety of routes to choose from. For example, although BGMP has to store all the non-aggregatable routes caused by multihoming, it can access a tree only at the root domain. On the other hand our protocol stores less routes while potentially maintains the ability to access the multicast tree at the nearest node.

- **Reduction in the necessary control traffic**

Multicast routes that are simultaneously unicast enabled get advertised for free. Only exclusive multicast routes generate separate advertisements. Although we still need to advertise the mapping between multicast groups and their root domains, this mapping is much more stable than the routes themselves and consumes little bandwidth to update.

- **Stability**

We showed above that BGMP incurs transient loops, which have an explosive effect and endanger the interdomain stability. Our protocol on the other hand ensures that no transient loops are possible since the tree is built correctly and never reshaped.

- **Variety of routes and Policies**

The search mechanism provides a wide space of useful routes to choose from, which enables a variety of policies and QoS, which would not be possible if routing and forwarding are coupled. In section 8.4.4, we have seen that path redundancy in the Internet is high, however many of the available useful paths are deserted only because of the coupling of routing and forwarding. By learning routes on demand we reallocate the resources available for multicast routes to reduce the amount of resources a domain spends on routes that might never be used, and devote them to generate a variety of routes to groups which the domain is interested in joining.

- **QoS amenable**

A QoS friendly architecture provides routes that satisfy the required quality of service and facilitates the establishment and maintenance of the necessary reservations. Our architecture provides a wide space of routes to a multicast destination, which increases the probability that certain requirements can be met. On the other hand, the pinned state adopted by the protocol helps maintaining the reservations and prevents them from becoming obsolete.

Chapter 8

Conclusion

8.1 Thesis Contribution

The first part of this thesis discusses a high-level design for an IP-anycast service. Our framework solves the intrinsic scalability problem associated with a global IP anycast service. To do so, it divides the Interdomain anycast routing system into two modules. The first module provides low-cost (free) sub-optimal anycast routes. The second module, run by the end domains, generates enhanced anycast routes that are customized according to the beneficiary domain's interests. The design incurs the usage of an adaptive caching system plus a route learning mechanism.

The second part of this thesis explores the benefits of the usage of the IP anycast service to extend the Core-Based-Tree multicast protocol (CBT) with the option to build optimized cost dynamic-core trees. These Trees are free from the two major drawbacks shown by CBTs, which are a poor-core-placement and traffic-concentration around the core router. The proposed extension makes all on-tree routers join the same anycast group such that a multicast tree appears to all off-tree routers as one network entity, which can be addressed directly. Consequently, data packets travel along the shortest path from source to tree consuming as little bandwidth as possible. Joins also make it to the tree along the shortest path, building short branches and consequently decreasing bandwidth consumption and join latency. Since the resulting tree is simultaneously an anycast group and a core based tree, we call it ACBT. Our simulation shows that ACBT reduces bandwidth consumption by a factor of 25%, and eliminates traffic concentration.

In the last part of this thesis, we developed FleSc -- a new interdomain multicast protocol that constructs bi-directional shared trees. Our protocol builds on the BGMP model to establish bi-directional trees rooted at an owner domain. However, our design differs from BGMP in two important ways. First, it uses ideas from our anycast framework to favor short local branches over hierarchical long ones. Second, the established tree uses hard state, which prevents transient loops and facilitates the establishment of alternative branches when the default ones do not accommodate the desired QoS. To scale, the architecture uses a small set of hierarchical aggregatable multicast routes. Each end domain extends these hierarchical routes with a set of on-demand generated routes that are customized according to the domain's interests. The on demand routes, which are generated using a mechanism similar to that used in YAM, provide a pool of possible branches to chose from the one that reduces bandwidth consumption or that satisfies certain QoS requirements. The design provides a mechanism to integrate such a branch in the tree without causing loops or starvation.

8.2 Future Work

Throughout this thesis we make a set of assumptions that, we think, are justified by the available data and the common sense. However, the experimental data is scarce, which makes it hard to accurately evaluate the assumptions. As a future work, we want to collect statistics about join/leave dynamics, multicast groups' topology, and multihoming topology. This will allow us to evaluate the efficiency of our constrained search and to estimate the exact effect of multihoming on scaling interdomain multicast.

The original idea of the Internet was to build a network that connects heterogeneous networking technologies by unifying the communication protocol used. Thus, no matter what is the underlying technology, applications can always expect the IP service to be uniform. As networks become a hot business, different providers will equip their networks with different services. Thus, the IP environment will not remain uniform. Definitely, the basic IP connectivity will be valid, such that old applications can get what they expect. However, the network will have many knobs that applications can tune to get better or different type of connectivity. We think that there will be a need for a protocol and a language that communicate to the application

its networking environment and the available services. In contrast to the service location protocol where the proposed services are end-to-end and the network's role is limited to discovering the servers and transferring this information to the interested clients, in this approach the services have a distributed nature and the networks' role is to build the service from its basic building blocks. For example, an application interested in exploiting the nearby available computational power would express its desire to the network. The network has to search the networking environment and return a graph that indicates the available free cycles and the type of connection they share. The application uses this graph to assign the different tasks to different machines according to the intensity of communication the tasks share. The resultant would be a parallel machine, whose architecture was optimized to fit the underlying network architecture,

Another follow-up research is exploring the use of anycast for load distribution among a set of replicated servers. By enforcing locality, anycast tends to balance the load among replicated servers. It creates around each server a dependency neighborhood, which relies on that particular server to provide the replicated service. If the replicated service is well provisioned then the dependency neighborhoods around the different servers will be evenly populated, and will impose an even load on the servers. However, occasionally a server would be overloaded, and it will like to shift some of the load to other nearby servers. We want to explore the possibility for reducing the size of an overloaded server's dependency neighborhood. To do so, the server asks its next hop router to advertise its anycast address with a larger metric. Thus, the server will appear to be further away, and some of the nodes at the edges of its dependency neighborhood shift their requests to nearby servers providing the same service (other members of the anycast group). In this context the following issues need study. First, assuming that no two close servers will try to shift off their load at the same time, can the system stabilize fast enough to rescue the overloaded server? What happens if two closely located servers try to shift off their loads simultaneously? Is it possible for the replicated servers to collaborate to shift the load off an overloaded server to a particular underloaded nearby replica?

Bibliography

Anycast References

1. S. Bhattacharjee, M. H. Ammar, E. W. Zegura, N. Shah, and Z. Fei, "Application Layer Anycasting." *in Procedings of Infocom*, Apr. 1997.
2. Z. Fei, S. Bhattacharjee, M. H. Ammar, and E. W. Zegura, "A Novel Server Technique for Improving the Response Time of a Replicated Service," *in Procedings of Infocom*, Apr. 1998.
3. C. Partridge, T. Mendez, and W. Milliken, "Host Anycasting Service," RFC1546, Nov. 1993.
4. E. Basturk, R. Haas, R. Engel, D. Kandlur, V. Peris, and D. Saha, "Using Network Layer Anycast for Load Distribution in the Internet," Aug. 1997.
5. J. Bound, P. Roque, "IPv6 Anycasting Service: Minimum Requirements for End Nodes".
6. R. Hinden, "Simple Internet Protocol Plus," White Paper, RFC1710, Oct. 1994.
7. P. Francis, "Pip Near-term Architecture" May 1994.
8. P. Francis, "A Call for An Internet Wide Host Proximity Service (HOPS)".
9. R. Hinden, S. Deering, "IP version 6 Address Architecture," Jul. 1997
10. D. Guyton, M. F. Schwartz, "Locating Nearby Copies of Replicated Internet Servers," Feb. 1995.
11. Y. Rekhter, T. Li, "An Architecture for IP Address Allocation with CIDR," RFC1518, Sep. 1993.
12. S. V. Fuller, T. Li, J. Yu, and K. Varadhan "Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation," RFC1519, Sep. 1993.
13. B. Carpenter, J. Crowcroft, and Y. Rekhter, "IPv4 Address Behavior Today," RFC2101, Feb. 1997.
14. J. Veizades, E. Guttman, C. Perkins, and S. Kaplan, "The Service Location Protocol," RFC 2165, June 1997.

15. D. Clark, J. Wroclawski, and K. Sollins, "Robust Multi-Scalable Networks," DARPA Proposal, MIT Laboratory for Computer Science, 1997.

Multicast References

16. S. Deering, "Multicast Routing in a Datagram Internetwork," Thesis Dissertation, Dec. 1991.
17. D. Wall, "Mechanisms for Broadcast and Selective Broadcast," Thesis Dissertation, Jun. 1980.
18. C. Shields, and J. J. Garcia-Luna-Aceves, " The HIP Protocol for Hierarchical Multicast Routing," 1998.
19. D. Waitzman, S. Deering, C. Partridge, "Distance Vector Multicast Routing Protocol," RFC1075, Nov. 1988.
20. A. Ballardie, "Core Based Trees (CBT) Multicast Routing Architecture," RFC 2201, Sep. 1997.
21. D. Estrin et al, "Protocol Independent Multicast (PIM) Sparse Mode" Working draft, 1996, ftp://netweb.usc.edu/pim.
22. W. Fenner; "Internet Group Management Protocol version 2 (IGMPv2)," Working draft, 1996.
23. T. Billhartz, J. Bibb Cain, E. Farrey-Goudreau, and D. Fieg, "A Comparison of CBT and PIM Via Simulation."
24. D. Estrin et al, "A Dynamic Bootstrap Mechanism for Rendezvous-based Multicast Routing," Technical Report, ftp://catarina.usc.edu/pim.
25. D. Estrin, M. Handley, S. Kumar, and D. Thaler, "The Multicast Address Set Claim Protocol," Work in Progress, Nov. 1997.
26. K. Almeroth, M. Ammar, "Characterization of the MBone Session Dynamics: Developing and Applying A Measurement Tool," Jun. 1995.
27. K. Calvert, E. Zegura, and M. Donahoo, "Core Selection Methods for Multicast Routing," 1995.
28. M. Donahoo, E. Zegura, "Core Migration for Dynamic Multicast Routing," Technical Report, 1995.

29. L.Wei, "Scalable Multicast Routing: Tree Types and Protocol Dynamics," Thesis Dissertation, Dec. 1995.
30. K. Carlberg and J. Crowcroft, "Building Shared trees Using One-To-Many Joining Mechanism," 1997.
31. M. Handley, J. Crowcroft, I. Wakeman, "Hierarchical Protocol Independent Multicast (HPIM,)" White Paper, Oct. 1995.
32. M. Doar and I. Leslie, "How Bad is Naïve Multicast routing," *In Proceeding of the IEEE Infocom*, 1993.
33. D. Thaler, D. Estrin, and D. Meyer, "Border Gateway Multicast Protocol (BGMP): Protocol Specification," Work in Progress, Mar. 1998.
34. D. Meyer, "Some Issues for an Inter-domain Multicast Routing Protocol," Work in Progress, Nov. 1997.
35. K. Almeroth and M. Ammar, " Collecting and Modeling the Join/Leave Behavior of Multicast Group Members in the Mbone," Aug. 1996.
36. M. Falousos, A. Banerjea, and R. Pankaj, " QoSMIC: Quality of Service Sensitive Multicast Internet Protocol," *In Proceeding of Sigcomm*, Sep. 1998.
37. W. Fenner, "Domain Wide Multicast Group Membership Reports," Work in Progress, Aug. 1998.
38. K. Kumar, P. Radoslavov, D. Thaler, C. Alaettinoglu, D. Estrin, and M. Handley, "The MASC/BGMP Architecture for Inter-Domain Multicast Routing", *In Proceedings of Sigcomm 98*, Sep. 1998.
39. C. Shields, "Ordered Core Based Trees," Master's Thesis, June 1996.

Routing References

40. Y. Rekhter, and B. Chinoy, "Injecting Inter-Autonomous System Routes into Intra-Autonomous System Routing: A Performance Analysis."
41. T. Bates, R. Chandra, D. Katz, and Y. Rekhter, "Multiprotocol Extensions for BGP-4," RFC 2283, Feb. 1998.

42. R. Govindan and A. Reddy, "An Analysis of Internet Inter-Domain Topology and Route Stability."
43. C. Labovitz, G. R. Malan, and F. Jahanian, "Internet Routing Instability," 1997.
44. I. Castineyra, N. Chiappa, and M. Steenstrap, "The Nimrod Architecture," RFC 1992, August 1996.

Simulation References

45. "Ns Notes and Documentation," The VINT Project.
46. M. Doar, "A Better Model for Generating Test Networks."
47. W. Zegura, K. Calvet, and S. Bahattacharjee, "How to Model an Inter-network", In the proceedings of Infocom, 1996.