Alternate Path Routing and Pinning for Interdomain Multicast Routing

(USC Computer Science Technical Report #97-655)

Daniel Zappala*
daniel@isi.edu
http://netweb.usc.edu/daniel
University of Southern California
Information Sciences Institute
4676 Admiralty Way, Floor 10
Marina del Rey, CA 90292-6695

Deborah Estrin†
estrin@usc.edu
http://netweb.usc.edu/estrin
Computer Science Department and
Information Sciences Institute
University of Southern California
Los Angeles, CA 90089-0781

Scott Shenker‡
shenker@parc.xerox.com
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304-1314

Abstract

Many researchers have explored enhancements of the Internet's best-effort service model that allow real-time and other inelastic applications to obtain preferential Quality of Service. However, these applications are limited to utilizing the opportunistic, shortest-path routes provided by the current routing infrastructure. To better support real-time applications, this paper introduces extensions to interdomain multicast routing to scalably compute and install alternate paths and non-opportunistic, or pinned, routes. We present a simple multicast setup protocol for installing alternate paths and discuss how it prevents loops. Furthermore, we include the results of a simulation study to demonstrate the viability of using localized route construction to find adequate alternate paths.

1 Introduction

The Internet has been extremely successful supporting elastic applications with best-effort service [Cla88]. However, best effort service can result in large and widely varying end-to-end packet delays if the links and routers traversed are heavily loaded. To better support real-time and other inelastic applications for which such vagaries are detrimental, the research community has proposed

*This material is based upon work supported by the Air Force Office of Scientific Research under Award No. DAAH04-94-G-0378.

†This material is based upon work supported by the National Science Foundation under Grant No. NCR-93-21043.

‡This research was supported in part by the Advanced Research Projects Agency, monitored by Fort Huachuca under contracts DABT63-94-C-0073. The views expressed here do not reflect the position or policy of the U.S. government.

extensions to the Internet's service model and architecture. These extensions would allow flows or flow aggregates to obtain preferential qualities of service (QoS) by marking packets or by using a "resource reservation" protocol.

While much research has been devoted to the development of admission control, scheduling, and queueing mechanisms, relatively little attention has been paid to upgrading the routing infrastructure of the Internet to support these extensions. Previous work [Bre95, EZL$^+$96] addressed some aspects of the problem for unicast routing. This paper is concerned with addressing the additional complexities that arise when considering multicast routing support for integrated services networks. Furthermore, we focus on interdomain routing, for which issues of scaling are more acute than for the intradomain case.

The routing infrastructure of the Internet has been designed primarily to support best-effort service. Current routing protocols [Hed88, Mil84, Hed89, IDR93, RL94, Moy94b] use opportunistic shortest-path routing for all applications. By *opportunistic* we mean that routing always utilizes the current shortest path, even if the previous shortest path is still functioning. By *shortest path* we mean that routing uses a single "cost" metric (often just hop-count) and then chooses the "least-cost" path.

In this paper, we focus on two particular problems that face real-time (or other performance-sensitive) applications that operate over current routing protocols:

- *Failed primary path problem*: Because current routing protocols use a single path, an application has no alternative route to try if it does not receive acceptable service along this path. This may happen, for example, if a flow does not achieve acceptable delay along an unreserved path, or if it attempts to make a resource reservation along the shortest path and is denied. Thus, many flows may be denied service even though other paths could accommodate their service requirements.

- *Opportunistic routing problem*: When current routing protocols adapt to a new shortest path, an application may experience a service disruption. For example, if a flow has obtained a good route, and then the route changes, portions of the new route may not have the necessary capacity. If the flow is unable to completely re-establish its desired service on the new path, service will be unnecessarily disrupted.

Many in the research community have proposed *QoS routing* techniques to solve these problems. A typical QoS routing scheme globally distributes topology, link resource availability, group membership, and (in some cases) per-flow resource usage. A source's first-hop router then uses this information to compute a multicast tree that is known a priori to have available resources. Finally, this same router uses a source-initiated setup protocol to install the multicast tree in the network. The bulk of this work attempts to minimize tree cost, primarily for static multicast groups [BKJ83, Wax88, Cho91, KPP92]. The Internet and ATM communities have begun applying these results to link-state multicast routing protocols [RGW97, ZSSC96, PNN].

Because these QoS routing approaches require global distribution and synchronization of such rapidly varying quantities, we do not believe they are applicable to interdomain routing, where issues of scale are paramount.[1] A global database of topology alone scales linearly with the size of the network; neither group membership nor the number of flows is limited by the size of the network. Additionally, the overhead required to maintain a consistent view of this data depends on application behavior, such as group membership changes and resource usage. We believe this

[1]QoS routing may still be used within a domain.

combination of storage and processing overhead rules out the use of centralized computation and installation of routes, as well as global optimization of these routes.

One alternative to QoS routing that does scale adequately for interdomain routing is to compute multiple paths for each destination based on relatively static routing metrics. In this approach, called *QoR routing*, the metrics reflect the "Quality of Route" using static service characteristics such as maximal bandwidth or minimal latency to indicate link capabilities. The routing protocol maintains a separate routing table for each metric, and applications indicate the their desired QoR when sending data. This is similar to the congestion-sensitive type-of-service routing described in [MS95], but routes adapt only to topology changes, not resource usage. QoR routing could provide significant benefits for best-effort service by allowing, for instance, interactive applications to avoid routes involving satellite links, while enabling applications involving asynchronous bulk data transfers to seek out maximal bandwidth paths. For similar reasons, QoR routing could benefit real-time and other inelastic applications.

Because QoR routing metrics are static, this approach has none of the scaling problems of the more dynamic QoS routing proposals. However, the essence of the failed primary route problem remains; an application could only avail itself of one minimal-latency route, and one maximal-bandwidth route, etc. If a service requirement was denied along one of these pre-computed routes, there would be no way of utilizing available bandwidth along other routes with similar properties. Thus, while QoR routing can increase the chance that an application will be satisfied with the primary route, receivers need routing to install *alternate paths* into a multicast tree on demand. Likewise, QoR routing does not solve the opportunistic routing problem, since QoR routes adapt as the metrics change. Therefore, receivers need routing to also install *pinned routes* – routes that will not adapt unless they fail – into a multicast tree on demand.

In this paper, we introduce a routing architecture in which alternate paths and pinned routes are installed by a multicast route setup mechanism. Existing route setup mechanisms use sender-oriented route setup or require routers to know the identities of downstream group members [DB95, FBZ94, UNI95], which renders them unusable for interdomain routing. We describe a simple, scalable route setup mechanism named MORF and show that it prevents loops while establishing and re-routing a multicast tree.

The key to the viability of this architecture is whether routers can find adequate alternate paths. To scale to the interdomain level, we propose to use route construction that is both decentralized and query-driven. Routers with local receivers find alternate paths for their receivers on-demand. Moreover, these routers do not use global distribution of topology to find routes. Rather, they find routes using a partial map of the network, which they build using heuristics to query the routing tables of other nodes. In this paper, we present the results of a simulation study demonstrating the viability of using localized route construction to find alternate paths around bottlenecks. Our intent in presenting these results is to demonstrate the utility of several low-cost, proof-of-concept route construction heuristics, thus validating our architecture. As others find better route construction heuristics, routers may incrementally deploy the improvements.

In our approach to route construction, we are designing for the case when congestion is not widespread, and thus do not attempt to find paths where resource availability is known a priori. We view this as the most feasible approach for interdomain routing for several reasons. During times of high load when congestion is common, using alternate paths can degrade network utilization [KZ89, Aki84]. Moreover, it is impracticable to design a scalable route computation method for finding the proverbial needle in a haystack — the one route that has available resources among a very large number of routes that do not.

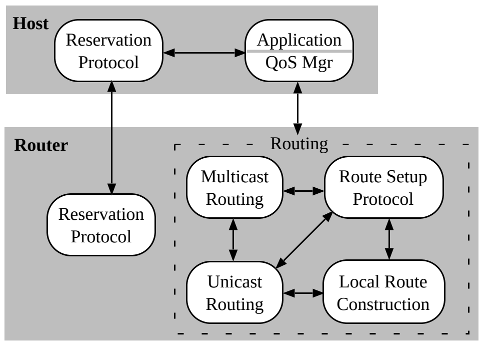


Figure 1: Routing Architecture

Thus, our solution to the failed primary route and opportunistic routing problems consists of two enhancements to the routing infrastructure: (1) a multicast route setup protocol and (2) localized route construction. We begin the rest of this paper by first discussing our multicast routing architecture in Section 2, showing how routing may install alternate paths and pinned routes on behalf of local receivers. Then in Section 3 we describe MORF, our multicast route setup protocol, and analyze its loop-freedom. In Section 4 we describe interdomain route construction heuristics we have developed and present simulations showing their effectiveness in finding alternate paths. Section 5 discusses related work and Section 6 presents our conclusions.

2 Multicast Routing Architecture

2.1 Overview

Our multicast routing architecture utilizes four primary components to build multicast trees (Figure 1). The multicast routing protocol constructs multicast trees based on routes obtained from the unicast routing protocol.[2] The route setup protocol installs alternate paths and pinned routes. It operates separately from the unicast and multicast routing protocols, but interacts with them to override their computations. Routers at the endpoints of the network (i.e. near hosts) may also incorporate a local route construction agent, which finds alternate paths for the route setup protocol.

Figure 1 also shows how applications and the reservation protocol interact with the routing architecture. Applications access route setup and reservation setup through two separate interfaces, in contrast to many QoS routing proposals where there is a single interface. Note that there is no application interface to the local route construction agent; the application's first-hop router contacts the agent when it needs a route. By not allowing the application to determine the routes being used, we prevent a malicious, malfunctioning, or misguided user from providing its own route and undermining the integrity or efficiency of a given tree.[3]

In the most simplistic model of how this architecture would be used, applications would interface directly with routing and the reservation protocol, respectively. In fact, given the complexity of the service options available, we assume that many operating systems will offer some form of support

[2]DVMRP [WPD88] uses its own internal unicast routing protocol, but this is equivalent for the purposes of our architecture.

[3]We are indebted to our colleagues Steve Deering and Van Jacobson for emphasizing that end systems should not control routing.

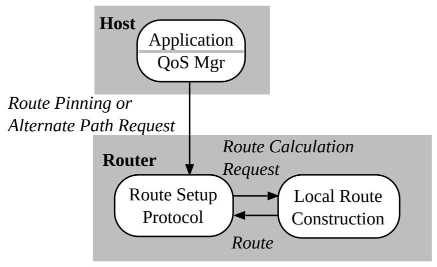


Figure 2: Example Use of Route Architecture

for managing quality of services. Such a *QoS manager* could act as an agent on behalf of an application, managing the reservation establishment and the routing services procurement process. A user (or a monitoring program) may simply indicate that service is unacceptable. The QoS manager could then choose from a number of actions, including asking for one of the available routing services, depending on the application. Thus, the QoS management function could reside either in the application itself, or in a QoS manager, or in some other form of operating system support. All of these possibilities fit within our proposed architecture; we are defining the services available to an end-host, not the organization of software on an end-host.

To initiate alternate path setup, an application or a QoS manager prompts routing for a different route (Figure 2). This signal may be prompted by many factors, including a user's unhappiness with packet delays or an admission control failure along the shortest path. The first-hop router then contacts the local route construction agent, which returns an explicit route[4] that meets the receiver's criteria. The setup protocol installs and pins this route, re-configuring the multicast tree at each hop.

To pin an existing opportunistic route, the application or QoS manager prompts routing to pin the current route (Figure 2). This signal may be prompted by any indication that the application is satisfied with the current route and wants to minimize its chances of disruption, for example if packet loss is low or it has succeeded in making a reservation along the route. The first-hop router probes the multicast tree to determine the current route, and encodes this route as an explicit route. The setup protocol then installs and pins this route, using the same mechanism as for an alternate path.[5] Note that the extra trip for probing the route allows routing to prevent loops during pinning by using an explicit route (see Section 3).

2.2 Multicast Route Setup Protocol

A number of reservation protocols perform route setup at the same time as installing a reservation [FBZ94, DB95, Sti95]. However, applications not using reservations may want to utilize route setup. For example, use of video- and voice-conferencing over the multicast backbone is widespread today [CD92], but each receiver is limited to using the shortest path. When congestion occurs on this path, a receiver may want to use route setup to install an alternate path, yet still use best-effort service over that path if it yields adequate performance. In this context, applications could use a reservation protocol as yet another, separate enhancement of this service model. This requires that

[4]We could use the term source route, but the route lists hops from the receiver to sender and is installed by a router near the receiver.

[5]In fact, a pinned route is really an alternate path in the sense that it is an alternative to the opportunistic route already in place.

we not embed route setup in the reservation protocol itself, but rather incorporate it into the basic routing infrastructure.

The function of the multicast route setup protocol, then, is to install explicit routes on behalf of receivers, overriding the opportunistic routes used by a multicast tree. Any route it installs is pinned so that it remains in place until it fails, at which time the multicast tree migrates back to an opportunistic route.

The setup protocol represents all routes as strict explicit routes, listing all hops in order from the receiver to the sender. This restriction enables the setup protocol to more easily guarantee the loop-freedom of the multicast tree (see Section 3).

2.3 Local Route Construction Agent

One of the key challenges of this multicast routing architecture is to develop a route construction protocol that may be applied to the interdomain scale. We distribute route computation to routers located near receivers and do not attempt to find routes based on link resource availability. Instead, routers first use the shortest path and then, as needed, find alternate paths that avoid any bottlenecks. This design follows the unified routing model [ERH92], in which commonly-used routes are pre-computed and routes used less frequently are computed on-demand.

Using localized route construction reduces the problem of constructing multicast routes to that of constructing unicast routes. Each route construction agent only needs to find a route between a local receiver and the sender, rather than having to take into account the entire multicast tree. This approach scales well to large multicast groups and allows agents to use existing unicast routing protocols as the basis for a route construction algorithm.

In addition, using localized route construction has several other advantages. First, because a route construction agent does not need to coordinate its routing decisions with other agents, it can utilize information that is not captured by the routing protocol's static metrics. This information can include the status of local reservation requests or resource availability information that is only known locally. In addition, route construction agents do not need to globally agree on a metric or algorithm. This allows diversity in the evolution of route construction techniques.

One consequence of using localized route construction is that routers may choose conflicting routes. Because any node in a multicast tree must have a single parent, routes using conflicting parents must be resolved. In the multicast routing architecture, the route setup protocol resolves such conflicts as it installs each route. Section 3 discusses this procedure in more detail.

3 The MORF Multicast Route Setup Protocol

We have designed the MORF multicast route setup protocol to install routes provided by local route construction agents. A router initiates installation of an explicit route by generating a Setup message containing the route (Table 1). MORF forwards the Setup message along the route, creating a Setup Tree that it maintains separately from the shortest-path tree built by the multicast routing protocol. Where the Setup Tree conflicts with the shortest-path tree, MORF overrides the shortest-path tree, and the multicast routing protocol prunes the conflicting branches (Figure 3a). MORF also adjusts forwarding table entries so that the resulting multicast tree reflects the path installed by MORF (Figure 3b). The multicast tree may be for a single sender [DEF+94], or multiple senders may rendezvous via a core [DEF+94, BFC93]. In either case, the protocol is the same; in the following discussion we refer to sender-based trees for simplicity.

Table 1: MORF Protocol Messages

Messages	**Parameters**		
Setup(*Group*, *Target*, *Route*)	*Group*	:	multicast group
Failure(*Group*, *Target*, *SetupRt*, *TreeRt*)	*Target*	:	sender or core
Teardown(*Group*, *Target*)	*Route*	:	explicit route
	SetupRt	:	route from Setup
	TreeRt	:	route used by tree

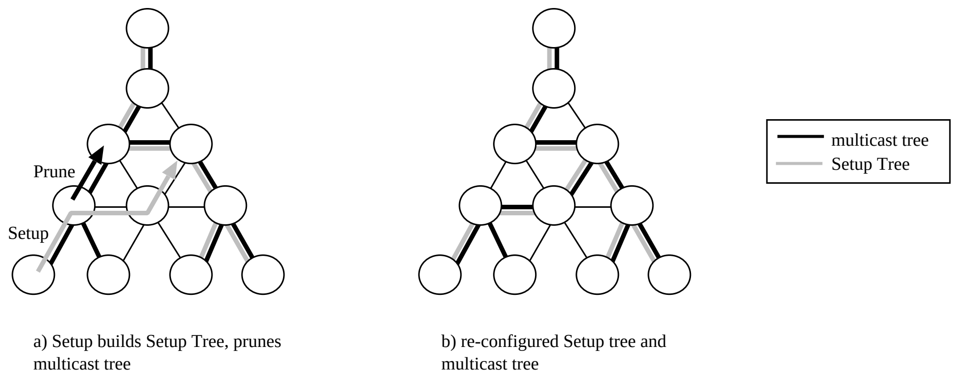
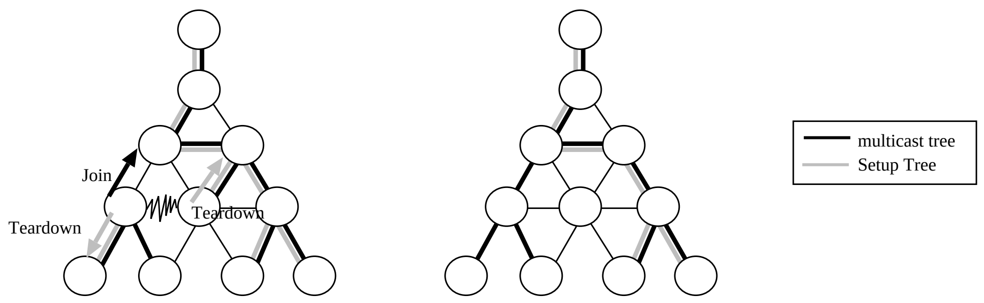

Prune
Setup
a) Setup builds Setup Tree, prunes multicast tree
b) re-configured Setup tree and multicast tree
multicast tree
Setup Tree

Figure 3: Using a Setup Message to Install a Route

Since the Setup Tree overrides default opportunistic routing, each router in the Setup Tree must have a mechanism to detect failures of an alternate path or a pinned route. The setup protocol may rely on a unicast routing protocol to exchange query messages with its neighbors to determine whether they are alive, or it may use its own similar mechanism. Once a failure is detected, MORF sends a Teardown message both upstream and downstream of the failure to remove failed branches from the Setup Tree (Figure 4a). At each hop, MORF notifies the multicast routing protocol of the branches it is removing. The multicast routing protocol re-builds the multicast tree to reflect its metric, often the shortest path to the sender (Figure 4b).

The above examples represent the simplified case when a Setup does not conflict with the rest of the Setup Tree. However, the setup protocol must also resolve Setup messages from different leaves that use conflicting routes, because leaf routers may use independent route construction agents. MORF resolves conflicts by choosing the first route that is installed for any given branch of the tree. Where subsequent routes meet this branch, they must conform to the route used from that point upward toward the source. If the setup protocol does not follow this restriction, then a number of looping scenarios may arise; Section 3.1 discusses these cases and the manner in which they are prevented.

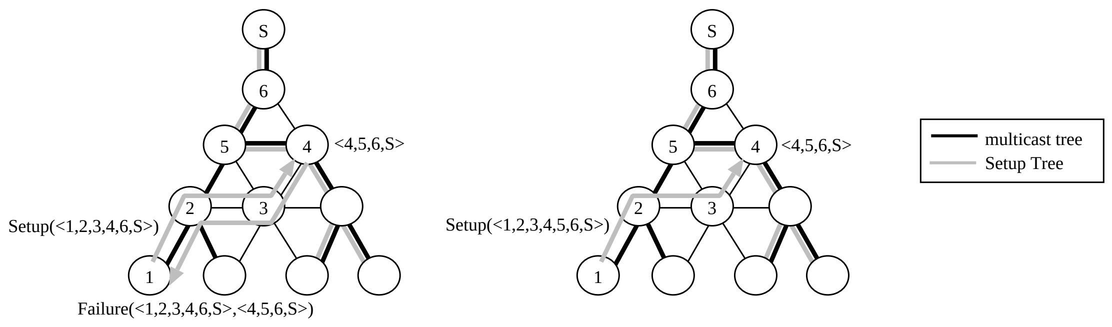
Figure 5 shows an example of how all Setup messages except the first one must match the route already used by the Setup Tree. When a Setup message adds a node to the Setup Tree, it caches the route it will use to travel from that node upward toward the sender. If a subsequent Setup message arrives at that node, it compares the remaining route it must travel to the route cached locally. If the routes do not match, the node stops processing the Setup and sends a Failure message downstream (Figure 5a). The Failure message contains the route used by the failed Setup and the



a) Teardowns remove AP-tree branches after failure, Join re-builds multicast tree

b) re-configured Setup Tree and multicast tree

Figure 4: Using a Teardown to Remove a Failed Route



a) Setup does not match, triggers Failure

b) Setup matches

Figure 5: Matching Setup Messages

route used by the tree from the rejecting node upward (Table 1). A router receiving a Failure message merges the two routes it contains to construct a new route that will match the tree, then sends a second Setup with this route (Figure 5b).

It is from this mechanism – *Match or Fail* – that MORF derives its name. By using this restriction, MORF may increase the setup latency, but it prevents any loops from forming while the tree is constructed. The remainder of this section discusses potential looping scenarios and analyzes the tradeoffs of MORF versus other potential solutions for preventing loops.

3.1 Preventing Loops

When Setup messages are not restricted to matching the rest of the Setup Tree, a number of possible looping scenarios arise. Figure 6a shows two Setups, each using an explicit route. Based on their order of arrival, as shown, if the Setups merge they form a loop. This loop can be prevented if nodes 1 and 3 compare the routes and detect the loop will form. However, when three joins are involved, as in Figure 6b, a single node cannot prevent the loop from forming without having more information available.

To prevent loops, a node can use one of two strategies:

1. Before adding a parent, the node checks all its descendants to be sure the parent is not already a descendant.

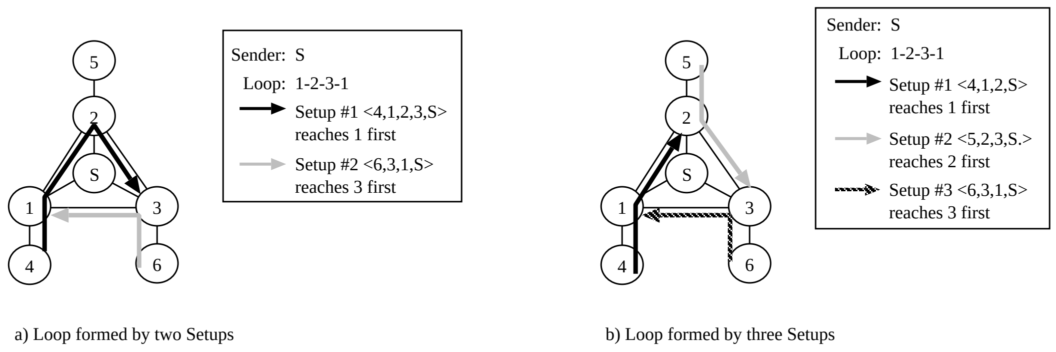

Sender: S
Loop: 1-2-3-1
Setup #1 <4,1,2,3,S> reaches 1 first
Setup #2 <6,3,1,S> reaches 3 first
a) Loop formed by two Setups
Sender: S
Loop: 1-2-3-1
Setup #1 <4,1,2,S> reaches 1 first
Setup #2 <5,2,3,S.> reaches 2 first
Setup #3 <6,3,1,S> reaches 3 first
b) Loop formed by three Setups

Figure 6: Loops Formed by Setup Messages

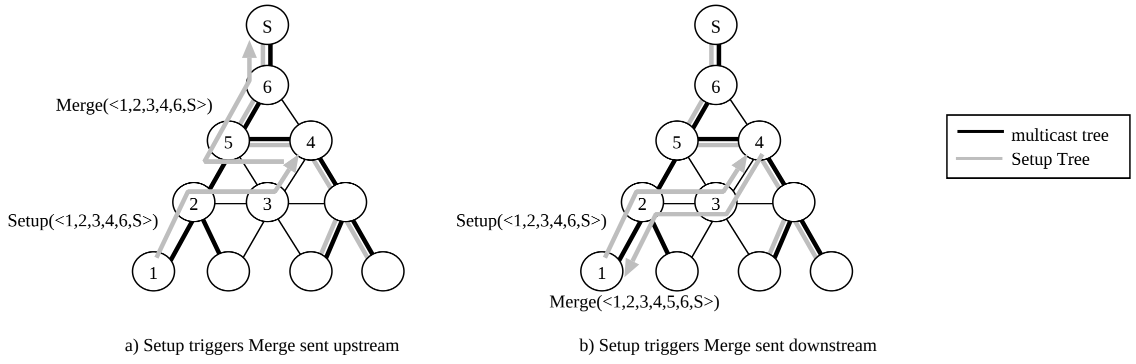


Figure 7: Merging Setup Messages Instead of Matching

2. Before adding a child, the node checks all its ancestors to be sure the new child is not already an ancestor.

We discuss each of these in turn. Due to the dynamic nature of multicast trees, a node may not know all of its ancestors or descendants. While a node knows the route embedded in the Setup message it has sent upstream, that message may have merged with another Setup carrying a different route. Likewise, other Setups may have merged downstream, adding new descendants.

One approach to keep nodes updated concerning upstream and downstream merges is to distribute information after each merge. Following solution (1) above, each Setup that merges can send a Merge message upstream containing its route (Figure 7a). Every node can then know all its descendants and thereby detect any loops. Alternatively, in keeping with solution (2) above, each Setup that merges can send a Merge message downstream containing the upstream portion of the route it merged with (Figure 7b). This allows every node to detect loops by knowing all its ancestors. We denote these two mechanisms as *Merge Up* and *Merge Down*, respectively. In both of these approaches, information distributed by the Merge messages may be stale, so loops such as that shown in Figure 6 may still form temporarily before being broken.

As opposed to these solutions, which in some cases will only detect loops after they have been formed, the strategy we use in MORF prevents any loops from forming. By requiring each Setup to match the upstream route already in place on the tree, MORF in effect enforces solution (2) by requiring that each node know its ancestors before it is added to the tree. Once a node is added to the multicast tree, its ancestors do not change. The cost of this strategy is that each Setup may

Table 2: Comparison of Setup Mechanisms

Mechanism Name	Message Overhead	Storage Overhead	Setup Latency	Loop Handling
MORF	O(1)	O(a)	1 or 3 trips	Prevent
Merge Down	O(1)	O(a)	1 trip	Detect/Break
Merge Up	O(d)	O(d)	1 trip	Detect/Break

take an extra roundtrip between itself and the rest of the tree.

Table 2 compares MORF to the Merge Down and Merge Up mechanisms, when building a single multicast tree, assuming there is no packet loss and that one receiver joins the tree at a time. The columns listing message and storage overhead consider the behavior of each mechanism at a single node. Overhead in these cases is expressed in terms of a, the number of ancestors of a node, or d, the number of descendants of a node. The setup latency column lists time in terms of the number of trips taken between a receiver and the multicast tree.

Clearly the Merge Up mechanism does not scale well because each node must store each descendent as well as send one message upstream for each descendant. The advantages of using a receiver-oriented mechanism are lost with Merge Up; a sender-oriented mechanism has the same message overhead, but only the sender must store the descendants.

The MORF and Merge Down mechanisms have a similar overhead in this situation. The MORF mechanism may have a longer setup latency, but in return has the distinct advantage that it may prevent rather than just detect loops, as discussed above.

3.2 Unicast Route Setup

Previous work has studied the efficacy of using explicit routing to support unicast real-time applications [Bre95]. One way to use explicit routes to provide alternate paths or pinned routes is to embed the explicit route in an application's packets [EZL$^+$96, HLFT94, DH95]. Assuming the route will be inserted by the sender's nearest router, no modifications to applications will be needed. However, because many routers currently delay processing of explicitly routed packets, this mechanism may not be applicable to applications with strict delay requirements.

An alternative is for the sender's nearest router to insert a label in the application's packets rather than an explicit route. This label would reference an alternate path or pinned route that is installed using MORF.[6] Because unicast applications involve only one receiver, the setup latency will only be 1 trip.

4 Interdomain Route Construction Heuristics

Given a scalable interdomain route setup protocol, the important issue we must address is whether we can construct useful alternate paths at a reasonable cost. Due to the scaling problems inherent in distributing global topology information at the interdomain level of the network, we propose

[6]The label could in fact be a multicast group address, reducing unicast alternate path routing to a special case of multicast.

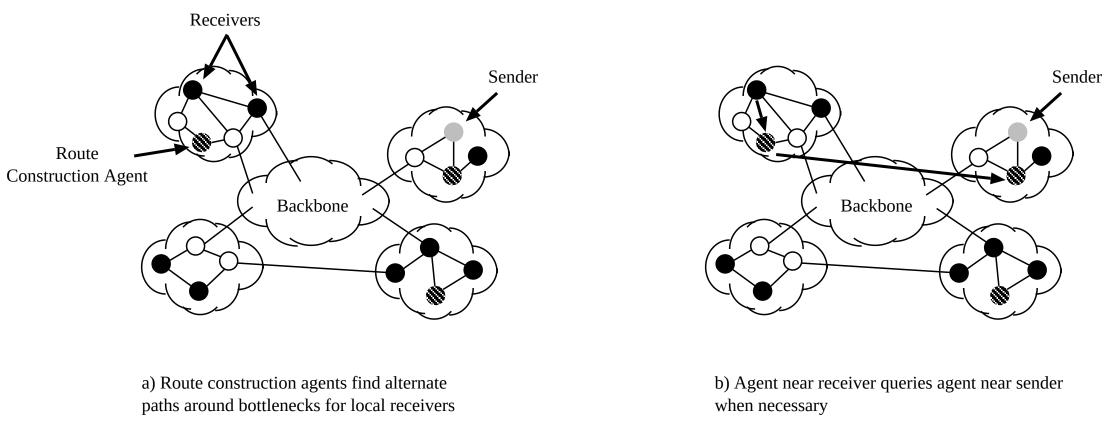

Receivers
Sender
Route Construction Agent
Backbone
a) Route construction agents find alternate paths around bottlenecks for local receivers
Sender
Backbone
b) Agent near receiver queries agent near sender when necessary

Figure 8: Route Construction Agents Serve Local Receivers

that a route construction agent use heuristics to partially explore the interdomain topology and find routes.

We have developed several low-cost, proof-of-concept heuristics that do not require changes to routing protocols, thus allowing incremental deployment at the edges of the network. To determine their effectiveness in finding alternate paths, we have conducted a simulation study over various types of random topologies. For the purposes of our simulations, we have concentrated on validating our approach by trying to find routes around a single overloaded interdomain-level link.

4.1 Approach

Our approach relies on route construction agents to serve local receivers (Figure 8a). When a receiver needs an alternate path, its local agent uses heuristics to find a route around any bottlenecks. If the local agent is unable to find an alternate path, it may contact an agent near the sender for a route (Figure 8b).

Because the agents do not have full topology information, they must build a partial map of the topology to find alternate paths. We have focused on methods for gathering topological information that is available with existing routing protocols. We have developed the following algorithm for exploring paths:

1. An agent explores the current path from itself to a small set of *Initial Nodes* in the network. These nodes may be randomly chosen or may consist of all the nodes within n hops. The agent initializes its map with these paths.

2. When the route setup protocol requests an alternate path from the agent, the request identifies a multicast tree (a group and a sender or core) and a bottleneck link. The agent probes the multicast tree for the requesting receiver's current path to the sender, adds this information to its map, and marks the bottleneck link.

3. The agent then runs a Dijkstra computation over its current map to find an alternate path around the bottleneck link. If one is found, it returns this path to the route setup protocol.

4. If a path is not found, the agent augments its map by exploring the current path from some other node in its map – a "third-party" – to the sender. The third-party is chosen using a

Table 3: Overhead and Path Length Bounds for Route Construction Heuristics

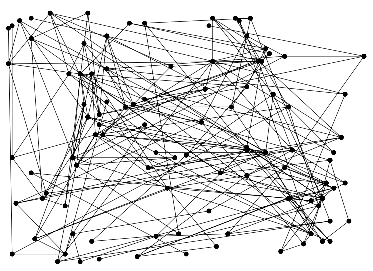
Heuristic	Overhead Bound
N-Hop	$c(c-1)^{N-1}$
N-Hop+Sender	$2c(c-1)^{N-1}$
M-Random	M
M-Random+Sender	$2M$
N-Hop+M-Random	$M + c(c-1)^{N-1}$
N-Hop+M-Random+Sender	$2M + 2c(c-1)^{N-1}$

breadth-first search of the map, starting from the agent, and finding the first *Initial Node* not already used as a third-party for the sender in question. Once a third-party is found that adds new links to the map, the agent returns to step 3 to re-run the Dijkstra computation. If no such node is found, then no alternate path can be found locally. At this point, the agent may optionally contact an agent that serves the sender of the multicast tree.

The variable parameters in this algorithm are the method by which nodes are selected to initialize the agent's map – either random or within n hops – and the option to query another agent near the sender of the tree when a route cannot be found locally. Combining the variations of these parameters yields the following set of heuristics:

N-Hop	Initialize using all nodes within N hops,
N-Hop+Sender	Same as above, but query the sender's agent if unable to find a route locally,
M-Random	Initialize using M random nodes,
M-Random+Sender	Same as above, but query the sender's agent if unable to find a route locally,
N-Hop+M-Random	Initialize using all nodes within N hops and M random nodes (not including any nodes within N hops),
N-Hop+M-Random+Sender	Same as above, but query the sender's agent if unable to find a route locally.

We can bound the overhead of each of the above hueristics in terms of the number of third-party queries performed for a single alternate path search. Table 3 lists these bounds in terms of c, the maximum degree of connectivity of the network; N, the number of hops explored initially; and M, the number of random nodes explored initially. In the algorithm given above, third-party queries are limited to the set of *Initial Nodes*, thus overhead is bounded by the size of this set. In the case of the N-Hop heuristic, this bound is $c(c-1)^{N-1}$; deriving the bounds for the other heuristics is straightforward. By keeping N small (i.e. 1 or 2 hops), we can limit the overhead of all of the heuristics to a small number of third-party queries.


a) 100-node flat random network

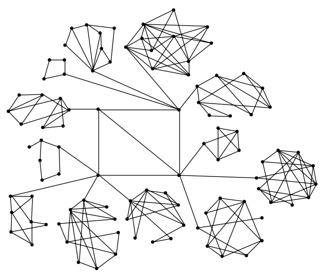

b) 100-node transit-stub network

Figure 9: Generated Networks

4.2 Simulation Model

We have implemented the route construction algorithm given above within the LBNL network simulator [MFF] to evaluate effectiveness in finding alternate paths. Our primary goal is to characterize the performance of the heuristics according to a varied set of topologies. We are also interested in measuring the path length of alternate paths computed using the heuristics; they should not be many hops longer than the shortest path.

4.2.1 Topology

We generated various large topologies using the Georgia Tech ITM topology generator [ZCB96, CZ]. We used one flat random network of 100 nodes (Figure 9a), using the Doar-Leslie edge-connection method [DL93] to generate edges that mostly connect nodes "near" each other. The average degree of connectivity for this network is 4.26. We also created transit-stub topologies, which consist of a backbone network and connected stub networks, with each sub-network generated randomly. Figure 9b shows a 100-node transit stub network we used, having an average degree of connectivity of 3.74. To determine how well our heuristics scale to larger topologies, we also generated three 1000-node transit-stub networks.

4.2.2 Workload

Given a single topology, the performance of the route construction heuristics will depend on the locations of the agent, the sender, and the bottleneck link. We have designed a workload that characterizes the effectiveness of a heuristic for a given topology by varying the placement of these three entities. For each topology, a simulation begins by randomly choosing a sender and a receiver. It then iterates through each of the links between the sender and receiver, assuming in turn that each link is the bottleneck link. The route construction agent, which we assume is co-located with the receiver, then tries to find an alternate path around that link.

The output of a single simulation, using a single sender-receiver pair, is the number of attempts made, the number of times an alternate path is found, the length of the alternate paths, and the number of route queries needed to compute the paths. If the heuristic is based on *M-Random*, then we repeat each simulation 100 times to compute averages for these numbers. Other heuristics only need to be run one time. For each of the generated topologies, we ran simulations for 100 sender-receiver pairs.

Table 4: Success Rate of Heuristics on 100-node Flat Random Network

Heuristic	Average Success Rate
1-Hop	0.77
1-Hop+Sender	0.93
1-Random	0.76
1-Random+Sender	0.91
1-Hop+1-Random	0.92
1-Hop+1-Random+Sender	0.98
2-Hop	0.97
2-Hop+Sender	1.00

4.3 Simulation Results

We ran a battery of simulations using the above workload for each of the route construction heuristics. We then evaluated each of the heuristics based on success rate and path length.

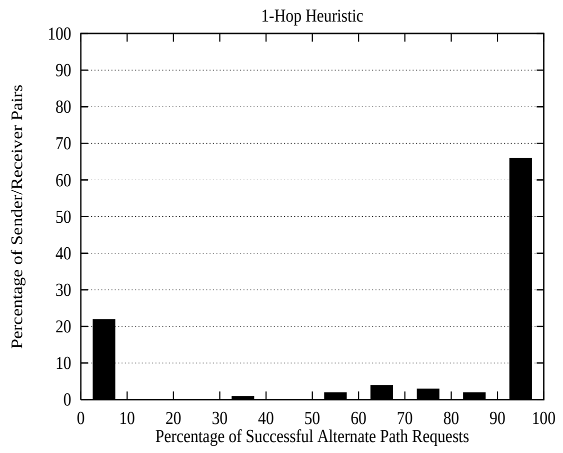
4.3.1 Success Rate

To determine the effectiveness of the heuristics, we compute the success rate by dividing the number of successes versus the number of attempts. We do not count attempts where there is no alternate path available, i.e. where even an algorithm with full knowledge of the topology will not find an alternate path. It is important to note that our experiments have underestimated the success rate because a given route server is finding alternate paths to only a small number of senders. In practice, a route server may handle requests for routes to a large number of senders, and any single request may benefit from routes learned for other requests. This will particularly be true when finding routes around local bottlenecks.

Table 4 shows the success rate of some of the route construction heuristics for a 100-node flat random network. For the flat random network, both the 1-Hop and 1-Random heuristics are able to find an alternate path about 75% of the time. Any combination of these heuristics with each other or with querying the sender is effective over 90% of the time.

Although the average success rate for the 1-Hop and 1-Random heuristics is nearly identical, the distribution of the success rate among various sender-receiver pairs is quite different. Figure 10 shows a histogram of the success rate for these two heuristics. For these histograms, we group the alternate path attempts of each sender-receiver pair and calculate the overall success rate of each pair. Based on these histograms, querying local routers will always be helpful for some sender-receiver pairs and will never help for others. On the contrary, querying a single random router will always help to find some alternate paths in a flat random network.

While the histograms are useful for determining the distribution of a single heuristic, graphing the cumulative distribution of the sender-receiver paired success rates is useful for comparing many different heuristics. For ease in reading these graphs, we have converted the cumulative distribution function $F_x(a) = P(x <= a)$ into a *diminutive distribution function* (DDF) $F_x(a) = P(x >= a)$. Thus, for a given point on the graph, the y value represents the percentage of sender-receiver pairs whose success rate is greater than or equal to the x value. For example, Figure 11a shows that, for the 1-Hop heuristic, 75% of the sender-receiver pairs find an alternate path 50% of the time, and



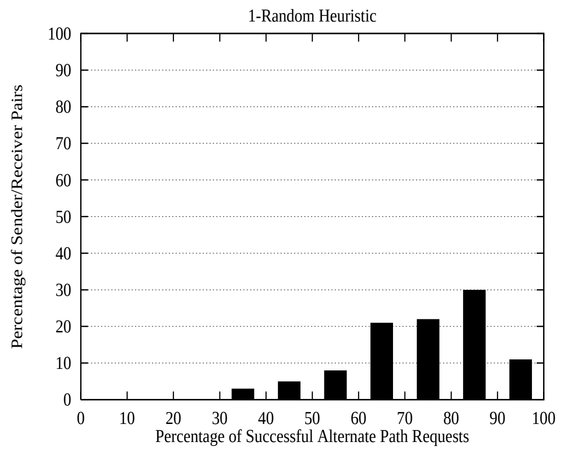
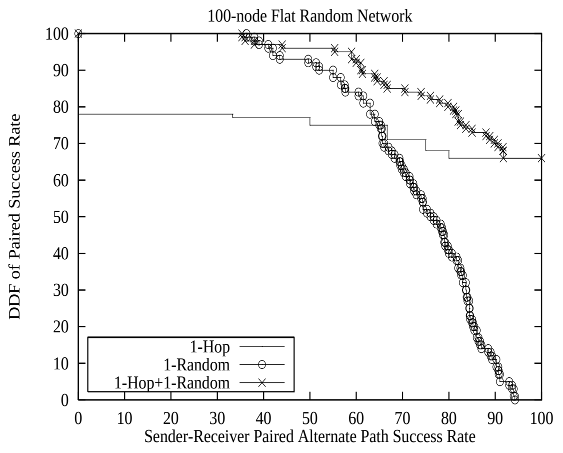
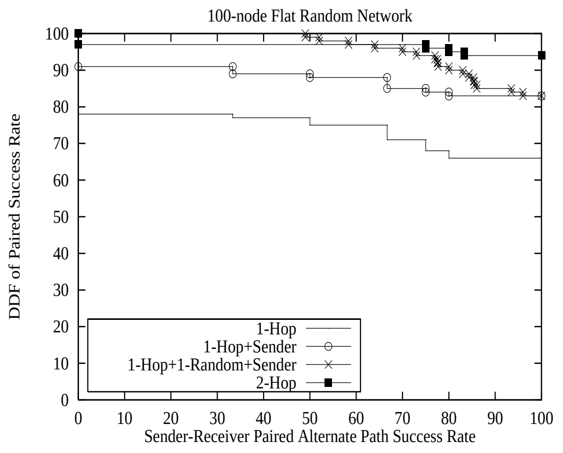


Figure 10: 1-Hop and 1-Random Heuristics on 100-Node Flat Random Network



a) Adding 1-Hop and 1-Random



b) Expanding from 1-Hop to 2-Hop

Figure 11: DDF of Heuristics on 100-Node Flat Random Network

66% always find an alternate path. This figure also shows that, for the 1-Random heuristic, all of the pairs are successful at least 36% of the time, but only 11% have a success rate higher than 90%. The 1-Hop+1-Random heuristic combines the potential for high success of 1-Hop with the lower bound of 1-Random.

Because 1-Hop has such a high potential success rate for a sender-receiver pair, we are interested in the value of combining it with other heuristics. Figure 11b compares the effectiveness of adding 1-Random and a query to the Sender to 1-Hop, versus expanding 1-Hop to 2-Hop. Adding just a query to the Sender to 1-Hop increases the percentage of pairs that always find an alternate path from 66% to 83%. Adding 1-Random to this combination raises the lower bound on success rate from 0 to 49%. While 2-Hop still has a lower bound of 0%, only 3% of the pairs fall in this category. Almost all of the other pairs always find an alternate path.

While many of the heuristics perform well on the flat random network, they all perform substantially worse on the 100-node transit-stub topology. Table 5 shows the success rate of the same

Table 5: Success Rate of Heuristics on 100-node Transit-Stub Network

Heuristic	Average Success Rate	Decline From Flat Random
1-Hop	0.35	0.42
1-Hop+Sender	0.68	0.25
1-Random	0.07	0.69
1-Random+Sender	0.15	0.76
1-Hop+1-Random	0.41	0.51
1-Hop+1-Random+Sender	0.74	0.18
2-Hop	0.41	0.56
2-Hop+Sender	0.77	0.23

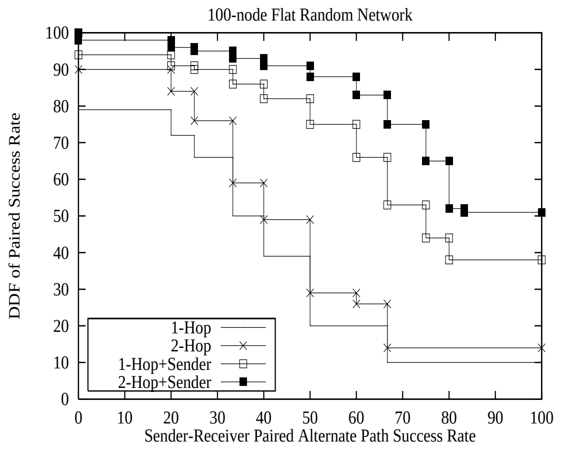


Figure 12: DDF of Heuristics on 100-Node Transit-Stub Network

heuristics run over the transit-stub network, along with the difference between this rate and that for the flat random network.

In particular, the 1-Random heuristic finds an alternate path only 7% of the time, compared to 76% for the flat random network. The reason for this decline can be seen by by examining the 100-node transit stub topology in Figure 9b. Nearly all of the nodes (96 out of 100) are located in one of the stub networks, each of which has an average of 8 nodes. Thus when the simulation randomly chooses a sender and a receiver, most likely they will be located within two different stubs. Then, when the agent randomly explores the path to one of the nodes, it is likely to choose a node that is in yet a third stub. With this map, the agent will not explore any part of the sender and receiver's stub networks except for the shortest paths through them. Thus the only times the agent is likely to find an alternate path will be the rare occasions when the third party is within the same stub as the sender or receiver or when the bottleneck is in the backbone.

The N-Hop+Sender heuristic, on the other hand, is better able to find alternate paths around local bottlenecks, either within the vicinity of the receiver or near the sender. Thus, this heuristic is much more successful on the transit-stub network than those using random nodes. Figure 12 demonstrates the effectiveness of querying the Sender in this topology, showing the DDF for 1-Hop and 2-Hop both with and without a query to the sender. Clearly the benefit of querying the sender

Table 6: Generation Parameters For Transit-Stub Networks

Network Name	Number Nodes	Number Transits	Transit Size	Stubs/ Transit	Stub Size
100	100	1	4	3	8
1000-1	1000	1	40	3	8
1000-2	1000	10	4	3	8
1000-3	1000	1	4	3	83

far outweighs the benefit of adding an extra hop to the heuristic. Likewise, increasing the number of hops has more impact when the heuristic also queries the sender, since both the sender and receiver are expanding the radius of their search. To further confirm the suitability of N-Hop+Sender for finding local bottlenecks, we re-analyzed the data for this topology considering only bottlenecks within a stub. In this scenario, the success rates of 1-Hop+Sender and 2-Hop+Sender rise to 84% and 92%, respectively.

Our simulations on 100 nodes thus confirm the following results:

- Topology affects the performance of the heuristics.
- Querying local nodes helps find alternate paths around local bottlenecks.
- Querying the sender always helps to find alternate paths, particularly for local bottlenecks near the sender.

We believe that finding routes around local bottlenecks is an important case because we expect the backbone of the network to be well-engineered, whereas a connecting domain may experience temporary overload.

To observe how well these heuristics scale to larger networks, we repeated our simulations using three 1000-node transit-stub networks. We generated each of these networks by building on the 100-node transit stub topology and making a different modification for each of the three networks. Table 6 lists the generation parameters for all of the transit-stub networks, which have an average degree of connectivity of 3.74, 4.35, 3.60, and 4.45 respectively. For Network 1000-1, we specified a larger transit network, resulting in a larger, highly-connected backbone and more stub networks (each node in the backbone retains the same number of average stub networks). For Network 1000-2, we specified more transit networks, resulting in a larger, hierarchical backbone and likewise more stubs. Finally, for Network 1000-3, we specified larger stubs, which retains a very small backbone and the same number of stubs. To keep the node degree low for these stubs, we used the Doar-Leslie edge connection method.

Table 7 shows the success rate of some of the heuristics run over these larger topologies as compared with the 100-node network. Because the 1-Hop+Sender and 2-Hop+Sender heuristics find alternate paths around local bottlenecks, they perform best on Network T1000-3, in which the size of the stubs dominate the network. Likewise, these same heuristics do not perform as well when the number of transit networks is increased in Network T1000-2. These two heuristics continue to find alternate paths around local bottlenecks, either within a stub or a transit network, but do not find alternate paths around distant bottlenecks when the connection between networks is hierarchical. On the other hand, when the backbone consists of a large, flat transit network, as

Table 7: Success Rate of Heuristics on 1000-node Transit-Stub Networks

	Average Success Rate			
Heuristic	Net T100	Net T1000-1	Net T1000-2	Net T1000-3
1-Hop+Sender	0.68	0.60	0.46	0.76
2-Hop+Sender	0.77	0.73	0.59	0.86
1-Hop+1-Random+Sender	0.74	0.90	0.83	0.80
1-Hop+Sender, Stubs	0.84	0.86	0.87	0.86
2-Hop+Sender, Stubs	0.92	0.97	0.97	0.97

Table 8: Path Length of Alternate Paths

	Average Number of Extra Hops per Alternate Path				
Heuristic	Net F100	Net T100	Net T1000-1	Net T1000-2	Net T1000-3
Global	0.67	0.69	0.62	0.79	0.63
1-Hop	0.81	0.64	0.62	0.70	0.61
1-Random	1.17	0.82	0.84	1.21	0.97
1-Hop+Sender	0.77	0.62	0.60	0.59	0.54
2-Hop+Sender	0.95	0.68	0.64	0.67	0.64
1-Hop+1-Random+Sender	0.97	0.64	0.69	0.96	0.57

with Network T1000-1, these heuristics can also find alternate paths within the backbone. The last two lines of the table emphasize the ability of the N-Hop+Sender heuristics to find routes around local bottlenecks. If overall performance on a variety of hierarchical networks is a consideration, then the 1-Hop+1-Random+Sender heuristic, by combining local queries with random queries, is able to consistently find routes around both local and distant bottlenecks.

4.3.2 Path Length

We measured the length of all of the alternate paths found by each heuristic and compared the length to the alternate paths found by the algorithm using global knowledge of the topology. The global algorithm finds the shortest available alternate path; thus, by taking the difference in path length we can determine the number of "extra" hops in the alternate paths found by the heuristics.

Table 8 lists the average number of extra hops for some of the heuristics on each of the topologies. We have grouped the data into several categories for ease in comparing the results. The first group lists the average number of extra hops for the global algorithm. Compared to this, the 1-Hop heuristic generally has comparable paths, with 1-Random having longer paths. The last group of data includes the three heuristics whose success rates are the highest. In almost all cases, the average number of extra hops is below one.

These results indicate that the heuristics often find an alternate path whose length is equal to that of the shortest path. If this also holds true for real-world networks, then a distance-vector unicast routing protocol like BGP [RL94] could pass through some equal-cost paths to route servers, simplifying route computation. If these paths are used frequently, then it will be cheaper to distribute them rather than compute them individually at each route server. A route server would still need to use the route querying heuristics described within this paper to find less-frequently used and slightly longer alternate paths when other paths are not adequate. This division of labor between pre-computed paths and on-demand computed paths has been proposed earlier as a part of the unified routing architecture [ERH92].

5 Related Work

The MORF protocol we present herein is novel in that it is receiver-oriented (making it scalable for multicast) and uses a simple mechanism to prevent loops. The ATM Forum has published a setup mechanism for multicast trees that includes receiver-oriented joins, but it requires the source to establish all branches of the multicast tree [UNI95].[7] In the Internet community, both the ST-II [DB95, DHHS93] and Tenet [FBZ94, BFG$^+$95] protocols establish multicast trees using primarily sender-oriented mechanisms. ST-II specifies additional mechanisms for receiver-oriented joins, but only along shortest-path routes and nodes in the tree must know the identity of all leaves in their subtree. More recent work allows receiver-oriented setup of non-opportunistic branches in a multicast tree [PGLA97], but again only along shortest-path routes and at the expense of considerably more state to prevent looping. Finally, Guerin et al. have recently proposed a sender-oriented route pinning mechanism [RG97], but the approach is applicable only to unicast routing and uses only shortest-path routes.

We have discussed several route construction heuristics that do not require global distribution of topology. Alaettinoglu explored route construction using aggregation and hierarchical heuristics for querying remote aggregates for more detailed information [Ala94]. Hotz has studied route construction heuristics based on route fragments and triangulation of graph position [Hot94]. In contrast to the above work, QoS routing protocols globally distribute topology and group information. A large body of QoS multicast routing literature uses Steiner trees to optimize tree cost for small, static groups [BKJ83, Wax88, Cho91, KPP92]. Salama et al. compare these and other approaches [SRV97]. The Internet and ATM communities have proposed multicast protocols using related techniques for more dynamic groups [RGW97, ZSSC96, PNN]. Because of scaling limits, this work is applicable for intradomain routing.

As a means for providing a variety of paths to meet application requirements, QoR routing is not a novel idea. Several routing protocols such as OSPF [Moy94b] allow the creation of multiple routing table entries per destination, and a Type of Service field in IP packets has been designed to index those tables [Alm92]. Recent work by Matta and Shankar [MS95] indicates that such an approach can improve overall end-to-end delays in the network. While their approach uses metrics based on measured delay and utilization, the formulations of their metrics result in relatively static measures that thus do not exhibit the oscillation seen in the ARPANET [KZ89].

The use of alternate path routing during periods of congestion is common in the telecommunications industry [AKK81, GKK88, MS91, HSS91, MG90, MGH91, AH93]. Trunk reservation is used to limit the use of alternate path routing under high load so that it does not decrease throughput [Aki84]. Likewise, the Internet community has explored the use of adaptive routing to avoid congestion and improve throughput [Att81, NSC90, WC90, Bre95]. Estrin et al. introduced the use of hop-by-hop routing for commonly-used routes, coupled with explicit routing for specialized routes [ERH92].

[7]Work in the academic community has provided more scalable receiver-oriented joins for ATM [BDG91], but does not provide the details of their protocol mechanisms.

Multicast routing in the Internet was pioneered by Deering, who developed procedures for hosts to notify routers of their group membership [Dee88a] and multicast routing methods for both distance-vector and link-state protocols [Dee91]. DVMRP [Dee88b, WPD88] is based on the flood-and-prune distance-vector method and MOSPF [Moy94a, Moy94c] extends OSPF to include group membership with link-state advertisements. PIM [DEF$^+$94, EFH$^+$97] introduces explicit join messages for sparsely-distributed groups and, along with CBT [BFC93, BJR95], allows creation of multicast trees that are shared by all senders. Comparisons of these protocols include [WE94, BCFG$^+$97].

6 Conclusions

We have developed two key enhancements for multicast routing that are needed for an integrated services architecture. First, we have designed a simple, loop-free, and scalable route setup protocol that routers can use to install alternate paths into a multicast tree on behalf of local receivers. This allows receivers to utilize routes other than the pre-computed shortest path and to install those routes as non-opportunistic paths. Second, we have affirmed that route servers can use localized route construction techniques to compute useful alternate paths without requiring global distribution of topology. We have demonstrated the utility of several low-cost, proof-of-concept heuristics; as others find better route construction heuristics, routers may incrementally deploy the improvements.

Because both the route setup and route construction heuristics are receiver-oriented, they may cooperate to reduce the impact of alternate paths on the network. We have shown that the route construction heuristics finds alternate paths whose average length is not much longer than that of shortest paths. If a computed alternate path matches what is already in the multicast tree, then its impact on overall tree size will be negligible. If it does not match, then the route setup mechanism will return the path in use by the multicast tree and require the originating router to modify its alternate path. Thus a router will always know the entire path from itself to the root of the tree, allowing it to check the impact a path will have on the tree before installing it.

7 Acknowledgments

Bob Braden and fellow researchers at ISI have encouraged and supported this work since its inception. John Heidemann helped to find an effective way of presenting simulation data. Ellen Zegura and Ken Calvert provided the GT-ITM topology generation software, along with excellent documentation and examples of its use.

References

[AH93] Gerald R. Ash and BaoSheng D. Huang. "An Analytical Model for Adaptive Routing Networks". *IEEE Transactions on Communications*, 41(11), November 1993.

[Aki84] J. M. Akinpelu. "The Overload Performance of Engineered Networks With Nonhierarchical and Hierarchical Routing". *AT&T Bell Laboratories Technical Journal*, 63(7), September 1984.

[AKK81] G. H. Ash, A. H. Kafker, and K. R. Krishnan. "Servicing and Real-Time Control of Networks with Dynamic Routing". *Bell System Technical Journal*, 60(8), October 1981.

[Ala94] Cengiz Alaettinoglu. *"Scalable Inter-Domain Routing with ToS, Policy and Topology Resolution"*. PhD thesis, University of Maryland, 1994.

[Alm92] P. Almquist. "Type of Service in the Internet Protocol Suite". RFC 1349, July 1992.

[Att81] R. Attar. "A Distributed Adaptive Multi-Path Routing - Consistent and Conflicting Decision Making". In *Fifth Annual Berkeley Workshop on Distributed Data Management and Computer Networks*, February 1981.

[BCFG$^+$97] Tom Billhartz, J. Bibb Cain, Ellen Farrey-Goudreau, Doug Fieg, and Stephen Gordon Batsell. "Performance and Resource Cost Comparisons for the CBT and PIM Multicast Routing Protocols". *IEEE Journal on Selected Areas in Communications*, 15(3), April 1997.

[BDG91] Rick Bubenik, John DeHart, and Mike Gaddis. "Multipoint Connection Management in High Speed Networks". In *IEEE INFOCOM*, 1991.

[BFC93] A. J. Ballardie, P.F. Francis, and J. Crowcroft. "Core Based Trees". In *ACM SIGCOMM*, August 1993.

[BFG$^+$95] R. Bettati, D. Ferrari, A. Gupta, W. Heffner, W. Howe, M. Moran, Q. Nguyen, and R. Yavatkar. "Connection Establishment for Multi-Party Real-Time Communication". In *Fifth International Workshop on Network and Operating System Support for Digital Audio and Video*, April 1995.

[BJR95] A. J. Ballardie, N. Jain, and S. Reeve. "Core Based Trees (CBT) Multicast: Protocol Specification". work in progress, November 1995.

[BKJ83] Kadaba Barath-Kumar and Jeffrey M. Jaffe. "Routing to Multiple Destinations in Computer Networks". *IEEE Transactions on Communications*, 31(3), March 1983.

[Bre95] Lee Breslau. *"Adaptive Source Routing of Real-Time Traffic in Integrated Services Networks"*. PhD thesis, University of Southern California, December 1995.

[CD92] S. Casner and S. Deering. "First IETF Internet Audiocast". *ACM SIGCOMM Computer Communication Review*, 22(3), July 1992.

[Cho91] C-H. Chou. "On Multicast Path Finding Algorithms". In *IEEE INFOCOM*, 1991.

[Cla88] David D. Clark. "The Design Philosophy of the DARPA Internet Protocols". In *ACM SIGCOMM*, August 1988.

[CZ] Ken Calvert and Ellen Zegura. "Georgia Tech Internetwork Topology Models". http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html.

[DB95] L. Delgrossi and L. Berger. "Internet Stream Protocol Version 2 (ST2): Protocol Specification Version ST2+". RFC 1819, August 1995.

[Dee88a] Stephen Deering. "Host Extensions for IP Multicasting". RFC 1054, May 1988.

[Dee88b] Stephen Deering. "Multicast Routing in Internetworks and Extended LANs". In *ACM SIGCOMM*, August 1988.

[Dee91] Stephen Deering. *"Multicast Routing in a Datagram Internetwork"*. PhD thesis, Stanford University, 1991.

[DEF+94] Stephen Deering, Deborah Estrin, Dino Farinacci, Van Jacobson, Ching-Gung Liu, and Liming Wei. "An Architecture for Wide-Area Multicast Routing". In *ACM SIGCOMM*, August 1994.

[DH95] S. Deering and R. Hinden. "Internet Protocol, Version 6 (IPv6) Specification". RFC 1883, December 1995.

[DHHS93] Luca Delgrossi, Ralf Guido Herrtwich, Frank Oliver Hoffmann, and Sibylle Schaller. "Receiver-Initiated Communication with ST-II". Technical Report 43.9314, IBM European Networking Center, 1993.

[DL93] Matthew Doar and Ian Leslie. "How Bad Is Naive Multicast Routing?". In *IEEE INFOCOM*, 1993.

[EFH+97] Deborah Estrin, Dino Farinacci, Ahmed Helmy, David Thaler, Stephen Deering, Mark Handley, Van Jacobson, Ching gung Liu, Puneet Sharma, and Liming Wei. "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification". work in progress, March 1997.

[ERH92] Deborah Estrin, Yakov Rekhter, and Steve Hotz. "A Scalable Inter-Domain Routing Architecture". In *ACM SIGCOMM*, August 1992.

[EZL+96] Deborah Estrin, Daniel Zappala, Tony Li, Yakov Rekhter, and Kannan Varadhan. "Source Demand Routing: Packet Format and Forwarding Specification (Version 1)". RFC 1940, May 1996.

[FBZ94] Domenico Ferrari, Anindo Banerjea, and Hui Zhang. "Network support for multimedia: A Discussion of the Tenet Approach". *Computer Networks and ISDN Systems*, 1994.

[GKK88] R. J. Gibbons, F. P. Kelley, and P. B. Key. "Dynamic Alternative Routing - Modelling and Behavior". In *Proceedings of the 12 International Teletraffic Congress*, June 1988.

[Hed88] C. Hedrick. "Routing Information Protocol". RFC 1058, STD 0034, June 1988.

[Hed89] Charles L. Hedrick. "An Introduction to IGRP". Technical report, The State University of New Jersey, October 1989.

[HLFT94] S. Hanks, T. Li, D. Farinacci, and P. Traina. "Generic Routing Encapsulation (GRE)". RFC 1701, October 1994.

[Hot94] Steven Hotz. *"Routing Information Organization to Support Scalable Interdomain Routing with Heterogeneous Path Requirements"*. PhD thesis, University of Southern California, 1994.

[HSS91] B. R. Hurley, C. J. R. Seidl, and W. F. Sewell. “A Survey of Dynamic Routing Methods for Circuit-Switched Traffic”. *IEEE Communications Magazine*, 25(9), September 1991.

[IDR93] International Standards Organization. *“Protocol for the Exchange of Inter-Domain Routing Information among Intermediate Systems to Support Forwarding of ISO 8473 PDUs”*, 1993.

[KPP92] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos. “Multicasting for Multimedia Applications”. In *IEEE INFOCOM*, 1992.

[KZ89] Atul Khanna and John Zinky. “The Revised ARPANET Routing Metric”. In *ACM SIGCOMM*, September 1989.

[MFF] Steve McCanne, Sally Floyd, and Kevin Fall. “LBNL Network Simulator”. http://ee.lbl.gov/ns.

[MG90] Debasis Mitra and Richard J. Gibbens. “State-Dependent Routing on Symmetric Loss Networks with Trunk Reservations: Analysis, Asymptotics, Optimal Design”. Technical Report 11212-900703-22, AT&T Bell Laboratories, July 1990.

[MGH91] Debasis Mitra, Richard J. Gibbens, and B. D. Huang. “Analysis and Optimal Design of Aggregated-Least-Busy-Alternative Routing on Symmetric Loss Networks with Trunk Reservation”. In *Proceedings of the 13th International Teletraffic Congress*, June 1991.

[Mil84] D. L. Mills. “Exterior Gateway Protocol for Formal Specification”. RFC 904, April 1984.

[Moy94a] J. Moy. “Multicast Extensions to OSPF”. RFC 1584, March 1994.

[Moy94b] J. Moy. “OSPF Version 2”. RFC 1583, March 1994.

[Moy94c] John Moy. “Multicast Routing Extensions for OSPF”. *Communications of the ACM*, 37(8), August 1994.

[MS91] Debasis Mitra and Judith Seery. “Comparative Evaluations of Randomized and Dynamic Routing Strategies for Circuit-Switched Networks”. *IEEE Transactions on Communications*, 39(1), January 1991.

[MS95] Ibrahim Matta and A. Udaya Shankar. “Type-of-Service Routing in Datagram Delivery Systems”. *IEEE Journal on Selected Areas in Communications*, 13(8), October 1995.

[NSC90] Don J. Nelson, Khalid Sayood, and Hao Chang. “An Extended Least-Hop Distributed Routing Algorithm”. *IEEE Transactions on Communications*, April 1990.

[PGLA97] Mehrdad Parsa and J. J. Garcia-Luna-Aceves. “A Protocol for Scalable Loop-Free Multicast Routing”. *IEEE Journal on Selected Areas in Communications”*, 15(3), April 1997.

[PNN] ATM Forum: PNNI SWG. *“PNNI Draft Specification 94-0471R13”*.

[RG97] S. Herzog R. Guerin, S. Kamat. “QoS Path Management with RSVP”. work in progress, March 1997.

[RGW97] A. Orda R. Guerin and D. Williams. "QoS Routing Mechanisms and OSPF Extensions". work in progress, March 1997.

[RL94] Y. Rekhter and T. Li. "A Border Gateway Protocol 4 (BGP-4)". RFC 1654, July 1994.

[SRV97] Hussein F. Salama, Douglas S. Reeves, and Yannis Viniotis. "Evaluation of Multicast Routing Algorithms for Real-Time Communication on High-Speed Networks". *IEEE Journal on Selected Areas in Communications*, 15(3), April 1997.

[Sti95] Burkhard Stiller. "A Survey of UNI Signalling Systems and Protocols for ATM Networks". *ACM SIGCOMM Computer Communication Review*, 25(2), April 1995.

[UNI95] ATM Forum: Technical Committee, Signalling Subworking Group. *"UNI Signalling 4.0"*, 1995.

[Wax88] Bernard M. Waxman. "Routing of Multipoint Connections". *IEEE Journal on Selected Areas in Communications*, 6(9), December 1988.

[WC90] Zheng Wang and Jon Crowcroft. "Shortest Path First with Emergency Exits". In *ACM SIGCOMM*, September 1990.

[WE94] Liming Wei and Deborah Estrin. "The Trade-offs of Multicast Trees and Algorithms". In *1994 International Conference on Computer Communications Networks*, September 1994.

[WPD88] D. Waitzman, C. Partridge, and S. Deering. "Distance Vector Multicast Routing Protocol". RFC 1075, November 1988.

[ZCB96] Ellen W. Zegura, Ken Calvert, and S. Bhattacharjee. "How to Model an Internetwork". In *IEEE INFOCOM*, 1996.

[ZSSC96] Z. Zhang, C. Sanchez, B. Salkewicz, and E. Crawley. "Quality of Service Extensions to OSPF". work in progress, June 1996.