A Scalable and Fault-Tolerant Architecture for Internet Multicasting Using Meshes*

J.J. Garcia-Luna-Aceves, Saravanan Balasubramaniyan, and Ramesh Balakrishnan
Department of Computer Engineering
University of California, Santa Cruz, CA 95064
jj,saro,ramesh@cse.ucsc.edu

ABSTRACT

The current architecture of IP multicasting is limited by the early binding of the membership of a router in a multicast group to the packet forwarding decisions the router must make for that group. We present a new architecture for IP multicasting that addresses these limitations by decoupling the mechanisms used for group addressing, group creation and management, and multipoint communication within a group. A new protocol is presented for the creation and management of multicast meshes, which substitute the traditional multicast trees as the underlying routing structure for multipoint communication within groups. Using simulation experiments, the overhead of mesh-multicast signaling and its packet-delivery ratios are compared against the overhead incurred with PIM-DM and CBT, which are well-known examples of tree-based multicasting in the Internet.

1. INTRODUCTION

The traditional IP multicast model established by Deering [6, 9] consists of three major architectural components: group addressing based on identifiers that are unique throughout the Internet, the separation of senders and receivers with anonymous host affiliation, and a tree-based routing structure. Today, the IP multicast architecture is the preferred approach to multipoint communication, and multicast routing trees (multicast trees for short) are used extensively for multicast routing in computer networks and Internet [1, 2, 7, 8, 15, 11].

The advantages of this multicasting model are efficiency and simplicity. However, even with all its attractive features, the IP multicast architecture is limited in its ability to adapt to either the dynamics of group interaction (members joining or leaving) or faults (routers and links failing). This stems from the early binding established between the decision a router makes to join a multicast group and the packet forwarding decisions the router must make to forward multicast packets addressed to the group. More specifically, when a router joins a multicast group, it must do so over the multicast routing tree that is used to forward multicast packets addressed to the group. Because the router should not create routing-table loops when it joins or re-joins a multicast routing tree, the joining mechanism is highly vulnerable to failures and the management of the multicast routing tree of a group is unnecessarily complex.

This paper proposes to decouple the mechanisms used for group creation and management from those used for multipoint communication within a group. This is achieved by using multicast meshes rather than trees as the basic routing structure and intra-mesh routing schemes. Section 2 summarizes our approach. The first proposals for multicast meshes were recently introduced within the context of ad-hoc networks based on broadcast links [4, 13, 12]. We show the first approach for establishing and maintaining multicast meshes efficiently in wired segments of the Internet, where routers have multiple interfaces and can interconnect through point-to-point or multipoint links. Section 3 outlines the *mesh administration protocol* (MAP) for the creation and maintenance of multicast meshes. Section 4 presents the results of comparing the signaling overhead and percentage of packet delivery of MAP with PIM-DM (protocol independent multicast, dense mode) [5] and CBT (core based tree) [2] using the $ns-2$ [22] simulator. The results of the simulation experiments clearly show that using meshes provides more fault-tolerant multicast delivery of data with far lower signaling overhead than using either shared multicast trees or per-source multicast trees.

2. COMPONENTS OF MESH-BASED MULTICASTING

Mesh-based multicasting assumes the availability of routing information from unicast routing protocols. The only requirement imposed on the routing protocols used is that they must provide correct distances to known destinations within a finite time. Mesh-based multicasting consists of the protocol to establish and maintain multicast meshes, and intra-mesh routing schemes.

A multicast group has one or multiple *anchors*, and each anchor of a group is a router in the multicast group that can be used as a landmark when joining the group. Globally unique IP multicast addresses can be used to identify multicast groups. Alternatively, one of many approaches based on local identifiers can be used (e.g., the address-pair approach first proposed in Simple Multicast [17]).

*This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) under Grant No. N66001-00-1-8942.

NGC '02 Boston, Massachusetts, USA

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2002		**00-00-2002 to 00-00-2002**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Scalable and Fault-Tolerant Architecture for Internet Multicasting Using Meshes	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Santa Cruz,Department of Computer Engineering,Santa Cruz,CA,95064**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**8**	

In mesh-based multicasting, rather than building and maintaining a multicast group by establishing one or multiple routing trees for the group, a *multicast mesh* is established for the group first, and multiple types of routing services are then provided within the mesh. A multicast mesh is a subset of the network topology that provides at least one path from each source to each receiver in the multicast group. The protocol used to establish and maintain the multicast mesh of a group attempts to keep the mesh connected and include all sources and receivers in the group as members of the mesh. A separate protocol defines how packets are forwarded within a mesh.

The *mesh administration protocol* (MAP) is used by routers to create and maintain a multicast mesh for each multicast group in which they need to participate. MAP extends the basic receiver-initiated approach for the creation of multicast trees introduced in the core-based tree (CBT) protocol [2] and used or refined in subsequent receiver-initiated multicast protocols (e.g., PIM-SM [8, 5] and BGMP [14]) to enable the creation of multicast meshes. First, an entity must create the multicast group, give it a name, select a set of anchors for the group, and assign an address to identify the multicast group. This information is made available in the Internet, so that sources and receivers can join the multicast group. A host obtains the address of the group and requests locally attached routers to join the multicast group by providing them with the group address.

A router can obtain anchor information for a given group in a number of ways. One approach consists of using similar mechanisms to those implemented for PIM SM, i.e., routers supporting multicasting must obtain the mappings between multicast group addresses and RP addresses. Another approach consists of extending the host joining mechanism to include the list of one or more anchors for the multicast group.

Unlike tree-based approaches for Internet multicasting, MAP does not predefine the paths that multicast packets take from sources to receivers of the group along the mesh. How IP multicast packets are forwarded within a mesh is determined by an *intra-mesh routing protocol*. The main objective of this protocol is avoiding any looping of multicast data packets within a mesh. An intra-mesh routing protocol can be based on information about distances from sources or the topology of the mesh; however, to maintain anonymity of the receiver set, a distance-vector solution may be preferable. The only requirement imposed on an intra-mesh routing protocol is that it must be loop-free at every instant, even when the unicast routing tables are inconsistent, in order to prevent routers from creating multiple replicas of multicast data packets that traverse such loops.

The single service provided by the current IP multicast architecture amounts to *group broadcast*, with which a packet from a source is sent on a best-effort basis to all group receivers. Accordingly, in the rest of this paper, the only intra-mesh routing that is assumed is intelligent flooding of each multicast data packet to all the routers in the mesh of a group. This is easily done by each router forwarding a packet that arrives over an interface in the mesh to all its other interfaces in the same mesh, provided that a copy of the same packet has not been forwarded by the router within a small time window [12]. This "packet-cache" approach is assumed in the rest of this paper.

3. MESH ADMINISTRATION PROTOCOL

MAP consists of a combination of receiver-initiated and sender-initiated mechanisms to maintain meshes. Receiver-initiated mechanisms are used to allow routers to join meshes, and sender-initiated mechanisms are used to improve the paths from sources to destinations within meshes. To describe MAP, we assume that the binding of a multicast group name to its address is accessible to any host and router, and that hosts communicate with routers through a proper protocol, such as IGMP [6, 10, 3], to request being added to or deleted from multicast groups. Furthermore, to simplify our description of MAP, we assume that all MAP messages sent by routers are transmitted reliably by means of an underlying protocol.

3.1 Information Maintained in MAP

A router i is required to maintain a unicast routing table (RT^i) built with the unicast routing protocol(s) used in the portion of the Internet where the router resides. For each destination j (i.e., an address range), RT^i must specify, as a minimum, the successor (s_j^i) and the distance to the destination (D_j^i). A router is also assumed to know the mapping between a group address and the address of at least one anchor for the group. MAP relies on the following additional data at router i:

- GMT^i : The *group membership table*, which maintains the known groups through each network attached to the router. An entry in the GMT specifies, for each attached network: (a) the address of a group, (b) a list of anchor addresses for the group, and (c) a flag indicating that hosts attached to the network have announced their membership to the group.

- NAT^i: The *neighbor affiliation table*, which specifies the groups in which one-hop neighbor routers and two-hop neighbor routers of router i participate, and whether any of its one-hop neighbor routers need router i to forward multicast data packets within a given group. For each one-hop and two-hop neighbor k, an entry in this table specifies: the address of a group and a forwarding flag set to 1 if the neighbor router requires router i to forward data for that group.

- $MILT^i$: The *mesh incident link table*, which specifies the cost of incident links with each of its neighbors. An entry in $MILT^i$ specifies the identifier of a neighbor k and the cost of the link from k to i, which we denote by l_i^k.

To update the above tables, routers exchange *MAP update messages*. Each message contains one or multiple entries and each entry specifies the entry type and its content. The entry type can specify a *mesh affiliation update*, a *join request*, or a *flush*. Each type of update entry is described subsequently.

The set of *mesh neighbors* of router i is defined as the set of neighbors of the router that are also members of the mesh for group g; this set is denoted by MN_g^i.

3.2 Sharing Mesh Affiliation Information

The primary objective of exchanging mesh affiliation information among routers is expediting the process of joining multicast meshes. Routers running MAP send to their immediate neighbors information about the constituency of

the mesh they know by means of *mesh affiliation updates*. These updates permit router i to know the multicast meshes in which its one-hop and two-hop neighbor routers participate. A *mesh affiliation update* from router i sent over a given interface specifies: (a) the address of the group, (b) a membership flag stating whether or not the router is a member of the group, and (c) the IP addresses of the one-hop neighbor routers of i that are members of the group.

When router i processes an input event that causes changes in its own mesh affiliations or the affiliations of its one-hop neighbor routers, it sends *mesh affiliation updates* for all the affected groups to all its neighbor routers. By allowing routers to know mesh affiliations for routers one and two hops away, the number of join requests that have to be sent towards group anchors is reduced, because routers may be able to join the mesh directly or send their *join requests* to routers known to be in the target multicast mesh, as we describe subsequently.

3.3 Basic Joining and Quitting Mechanisms

Joining a multicast mesh is a receiver-initiated process that differs from the approach introduced with CBT in that: (a) routers are added to the routing structure of a group through shortest paths from the group to the new member routers, rather than through reverse shortest paths; and (b) a router need not send *join requests* to join a group, because of the mesh affiliation information exchanged among routers. With respect to its membership in a multicast mesh, router i is said to be

- *Idle,* if it is not a member of the mesh;
- *Member,* if it is a member of the mesh;
- *Relay,* if the router is a not a member of the mesh but is required to build the mesh.

The basic joining mechanism ensures that the shortest path between the anchor or *member* router that responds to a *join request* and the origin of *join request* is part of the mesh. A router may join a multicast group directly through one or more of its neighbors, or it may send a *join request* towards an anchor of the group. If the router has one-hop neighbor routers in the target mesh, the router can just join through all its interfaces to those neighbors. If the router has two-hop neighbors in the target mesh, it can send *join requests* towards those routers. If the router has no one-hop or two-hop neighbors in the target mesh, it sends a join request towards the nearest anchor of the group.

When a *member* router in the target mesh receives a *join request*, it sends a *mesh build* to the neighbor router in the shortest path to the origin of the *join request*. A router that receives a *mesh build* becomes part of the mesh, changes its state to *relay* and forwards the *mesh build* along the shortest path towards the origin of the *join request*. When a *relay* router in the target mesh receives a *join request*, it forwards the *join request* towards the nearest member router known to it. If there are no member routers that are within two hops of the *relay* router it simply forwards the *join request* along the mesh to its parent (upstream router that forwarded the *mesh build* to this router). This process continues until the *join request* reaches an anchor or a member router.

The content of a *join request* sent by router i is a tuple (k, *gid*, o, *hint*) specifying the intended relay router (k), a group address (*gid*) identifying the group to be joined, the origin of the *join request* (o), and a hint consisting of a list of one or more anchor addresses. A *join request* is forwarded on a hop-by-hop, best-effort basis towards anchors or routers.

The content of a *mesh build* from i is a tuple (k, *gid*, o) specifying the intended relay (k), a multicast group identified by (*gid*), the origin of the *join request* that created the *mesh build* (o). A *mesh build* is forwarded on a hop-by-hop, best-effort basis along the shortest path towards the origin o of the request that created the expansion of the mesh.

If router i is *idle* and requires to join a multicast group *gid* because of a request from an attached host and some of its one-hop or two-hop neighbors have *member* state, it sends the *join request* (n, *gid*, $o = i$, *hint*) to the unicast next-hop towards the *member* neighbor. If router i has no one-hop or two-hop neighbors that are group members of the target group *gid*, then it sends the join request (n, *gid*, $o = i$, *hint*) to the next-hop router n along the shortest path to the nearest anchor of the target group *gid*.

When router i receives a *join request* from a neighbor router k consisting of the tuple (i, *gid*, o, *hint*), it acts according to its own state and the state of its neighbors as follows:

- If i has *member* state for the target group *gid*, it replies to i with the expand (k, *gid*, o), where *gid* and o are the same as in the *join request* that causes the expand.
- If i is in the *idle* state for the target group *gid*, has one-hop or two-hop neighbor routers with *member* state in *gid*, it sends the join request (n, *gid*, o, *hint*) to the one-hop *member* neighbor n or relays the *join request* to the neighbor n that is the unicast next-hop to the two-hop neighbor with *member* state in *gid*.
- If i is *idle* for the target group *gid* and has no one-hop or two-hop neighbors with *member* state in the group, then it relays the *join request* (n, *gid*, o, *hint*) to the unicast next-hop (n) along the shortest path to an anchor of *gid*.

For a router, leaving a group is as simple as announcing that it stops being part of the corresponding mesh. Router i announces that it stops being a member of the mesh for group *gid*. The neighbors update this information when they receive the *flush* message from the router that left the group. Router i can leave group *gid* only if and only of it is in *member* state, has no attached hosts that are members of *gid*, and none of its neighbors need router i to forward data for the group.

3.4 Mesh Building Mechanism

The basic joining mechanism ensures that the shortest path between the anchor or *member* router that responds to a *join request* and the origin of *join request* is part of the mesh. This is achieved when the anchor or *member* router sends a *mesh build* that makes all the routers lying in the shortest path between the source of the *mesh build* to the origin of the *join request* as *relay*'s of the mesh. In addition to the basic mesh joining and quitting mechanisms, MAP implements a couple of heuristics to build more than just a shared multicast tree, without involving too many routers in the process.

To build more than a multicast tree spanning all the receivers of a group, receivers have to be given at least two

paths to the rest of the mesh. MAP attempts to provide an alternate path that is node disjoint to the shortest path with which a receiver is linked to the mesh. The heuristic employed is very simple and is based on an expanding ring search that tracks the set of routers used by a receiver to join a multicast mesh. A router that joined the multicast mesh as a new member, through the shortest path between the mesh and the new member, sends an acknowledgment to the *member* router on the mesh that initiated the *mesh build*. The acknowledgment captures the identifier of all the routers on the shortest path that connected the mesh and the new member. The *mesh build* initiator builds a *prohibited list* from the list of nodes in the acknowledgment and then attempts to build an alternate path to the new member that is completely node disjoint with the shortest path. The *mesh build* initiator sends a *mesh request* to all its appropriate one-hop neighbors(routers whose identifier is not in the *prohibited list*) along with the *prohibited list* requesting them to find an alternate path to the new member without using any routers listed in the *prohibited list*.

Any router that receives a *mesh request* forwards the *mesh request* to its unicast next-hop towards the new member after appending its identifier to the *prohibited list* and checking that the path-length criterion for the alternate path still holds.

If a router that received a *mesh request* does not have an appropriate next-hop towards the new member (i.e., if its unicast next-hop is on *prohibited list*), it performs a distance query, whereby it queries all its appropriate one-hop neighbors of their respective distances to the destination and chooses to forward the *mesh request* to the neighbor that reported the least distance. If no appropriate neighbor exists or if the path-length criterion is not satisfied, then it sends a negative acknowledgment to the router that forwarded the *mesh request* to it.

When a router receives a negative acknowledgment, it first checks if it had already done a distance query. If it had not done one already, it proceeds to do a distance query to all its appropriate neighbors. If it had done a distance query already, then it selects the router that reported the next best distance to the destination router. This way, a router that receives a *mesh request* exhausts all its possible ways to find an alternate path to the desired destination. When the *mesh request* reaches the destination router i, it sends a positive acknowledgment that traces its way back to the initiator of the *mesh request* along the new alternate path and carries the identifiers of the routers that formed the alternate path using which the *mesh request* initiator builds the new alternate path. If more than one alternate path is present, the *mesh request* initiator chooses the shortest alternate path reported.

Note that our mesh building mechanism does not flood the network and the query messages travel only to 1.5 times the shortest path-length number of hops. The major challenge in building the mesh is caused by the unicast routing table, which does not give alternate paths and omits paths to a destination that are as good as the shortest paths. The mesh building mechanism circumvents this limitation posed by sending distance queries.

To reduce the number of routers that belong to a multicast mesh unnecessarily, relay routers on an alternate path ensure that they are connected to the mesh by only one upstream link. Hence, a relay router sends a *flush* to the upstream link that connected the router to the mesh before a new (shorter) alternate path is established to the mesh. If a relay router that is on the shortest path between two member routers receives a *mesh build* from a new router, it does not send a *flush* to its upstream router, because this would disrupt the shortest path to which it belongs.

3.5 Handling Topology Changes

A router's reaction to link or node failure depends on the status of the router in the mesh. A *relay* router sends a *flush* message to all the links through which it is connected to the mesh if it detects that it has fewer than two connections to the mesh. A *flush* message relieves the state of the router from *relay* to *idle* and is forwarded by each *relay* router to all its neighbors that participate in the mesh.

A *flush* travels along the mesh making all the *relay* routers in its path to become *idle*. When a *member* router receives the *flush*, its reaction to the *flush* depends on the router that sent the *flush*. If the *relay* router was its parent router on its shortest path to the mesh, the *member* router has to join the mesh following the mechanism it used when it had to join the mesh for the first time. However, once the *member* router gets the *mesh build* as part of the mesh rebuilding mechanism, it sends a *flush* to the parent router that connects it to the mesh in its alternate path to remove the old alternate path.

The old alternate path is removed because once the shortest path is broken and a new one is built there is no way for the destination router to which the new path is built to know if the new shortest path overlapped with the old alternate path. When a *member* router gets a *flush* from the parent router of the alternate path, it simply requests the upstream *member* router or anchor, whichever is the closest, to build an alternate path to it. *Flush* messages from routers that are downstream (child) are ignored as it is the responsibility of the downstream *member* routers to inform the upstream *member* routers to build the mesh when it is broken. The mechanism is slightly different for the case of a router that is a sender, in that the sender router is responsible for building the mesh towards the anchor or nearest *member* router when it detects some node or link failures that affect its connection to the mesh.

3.6 Keeping Meshes Connected

A multicast mesh constructed for a group may become partitioned due to very high link or node failure rate. In such cases, MAP allows new routers to join the mesh by using multiple anchors for a group. Meshes built around various anchors should be connected with one another to facilitate communication of data throughout the multicast group. To facilitate this, each anchor builds a mesh to two of its closest anchors if there are more than two anchors. In order to keep all the anchors connected to one another, an anchor builds a mesh to an anchor only if its address is smaller than the address of the anchor to which it is building a mesh. Hence, for a group with multiple anchors, this scheme ensures that all anchors are connected to each other and that each anchor can reach any other anchor. The mesh building proceeds just as in the case of building a mesh towards new member routers.

3.7 An Example of Mesh Building

Fig. 1 shows an example of the mesh building mechanism.

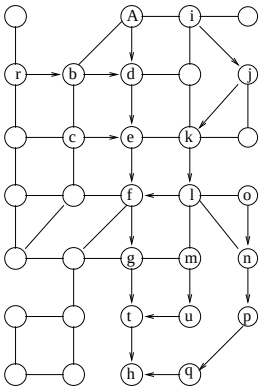

Figure 1: An example to illustrate Mesh Building using MAP

Router h wants to become member of a multicast group for which router A is the anchor. Links with arrow marks indicate the unicast next-hop router that the router at the tail of the arrow uses the router at the head of the arrow to reach router h. Router h sends a *join request* to router A, which sends *mesh build* to router h making all the intermediate routers d, e, f, g, t, between A and h to become *relays*. Thus the shortest path between A and h becomes part of the mesh.

When router h receives the *mesh build*, it sends an acknowledgment addressed to router A, the initiator of *mesh build*, to router t which had forwarded the mesh build to it. The acknowledgment carries the path information(t,g, f, e, d) between router h and router A, that is made part of the mesh by carrying the stamps of router identifiers through which it passes before reaching router A.

Router A constructs the *prohibited list* of routers using the path information contained in the acknowledgment and sends a *mesh request* to all its one-hop neighbors b, i except the one on the *prohibited list* (router d). The *mesh request* carries the *prohibited list* so that the receiving routers can build an alternate path that is disjoint with the shortest path to h. Router A also specifies the maximum number of hops to which the search for alternate path can proceed to and sets it to 9 distance units (which is 1.5 times the shortest path-length).

When router b receives the *mesh request* it finds that its unicast next-hop to router h is router d which is already on the *prohibited list*. Hence router b adds its router identifier to the *prohibited list* and sends distance query to all its appropriate one-hop neighbors (c,r). However, since the respective next-hops to router h for both c and r are already on the *prohibited list*, they reply with a distance of infinity to router b. Since router b does not have any other option to search for the alternate path it sends a negative acknowledgment to router A, which sent the *mesh request* to it. When router i receives the *mesh request* from router A, it forwards the *mesh request* to its appropriate next-hop router j which in turn forwards it to router k as router j uses router k as the next-hop to reach router h. Note that the three routers router i, j, k that receive *mesh request* are appropriate routers and their router identifiers are added to the *prohibited list* when they forward the *mesh request*. Router k forwards the *mesh request* to its next-hop router l.

After receiving the *mesh request*, router l discovers that its next-hop to router h as indicated by its unicast routing table is router f, which is on the prohibited list. It proceeds to perform a distance query to all its appropriate one-hop neighbors, m, o and n. Router m reports a distance of 3 distance units, router n reports 4 distance units and router o reports 5 distance units. Router l chooses router m as it reports the least distance to the destination and forwards the *mesh request* to router m. Router m forwards the *mesh request* to router u. Router u's next-hop to h is t which is on the *prohibited list* and hence u sends a negative acknowledgment to router m which sends it to l after finding that it does not have any appropriate neighbor through which it can continue the search process to find the alternate path. Router l after receiving the negative acknowledgment from router m proceeds to continue the search process by forwarding the *mesh request* to router n which reported the next best distance to router h when it conducted distance query. The *mesh request* forwarded to router n contains the list of all routers which took part in the search process initiated by router i. Router n forwards the *mesh request* to its next-hop router p which in turn forwards the *mesh request* to router q. Router q forwards the *mesh request* to router h which sends an acknowledgment addressed to router A, the initiator of *mesh request*, to router q, the router that sent it the *mesh request*. Each router that receives the acknowledgment forwards it to the router that had earlier sent it the *mesh request*. In this way the acknowledgment reaches router A carrying the stamp of the router's identifier through which it passes. Router A constructs the alternate path using the acknowledgment and sends a *mesh build* through this path making all the intermediate routers between router A and h as *relays*.

3.8 Gratuitous Reply

Our simulation experiments have shown that most of the searches for alternate paths fail, because of the nature of the IP unicast routing table, which is a single-path table. We illustrate this with an example for a simple topology shown in Fig. 2.

The links with an arrow head in Fig. 2 indicate the next-hop router used to reach router i by the router at the tail of the arrow. Router j builds the mesh towards router i and incorporates routers k and l into the mesh as they form the shortest path between router j and i. Router j sends a *mesh request* to router m as part of searching for the alternate path, which in turn is forwarded to router n. Router n performs a distance query to router o as its next-hop to router i is router k which is on the *prohibited list*. Router o would report a distance of infinity if it did not use the gratuitous reply mechanism as its next-hop to router i is router l which is on the *prohibited list*. Due to the use of the gratuitous reply mechanism, it checks the distance of its neighbor router p and since the distance of router p is lesser than its own distance it concludes that router p is using a different route to reach router i and hence reports router p's distance added with its distance to router p as the reply to the distance query from router n. Hence every router that is about to report infinity in its reply to a distance query checks if any of its neighbors distance to the desired destination is less than or equal to its own distance and if one such exists it reports that distance. In this way, the gratuitous reply mechanism helps to find the alternate paths

that may not have been possible to find due to the nature of the unicast routing table. This mechanism is used only when the *mesh request* reaches the destination within 3 hops, otherwise excessive overhead (in terms of search queries) would be generated.

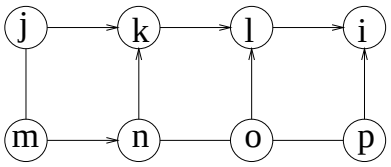

Figure 2: Figure explaining the gratuitous reply

4. PERFORMANCE OF MAP

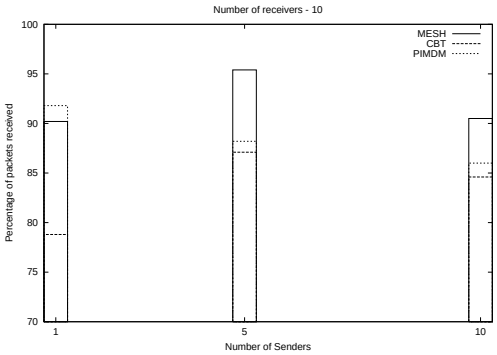

Figure 3: Percentage packet delivery (R-Receivers, S-Senders)

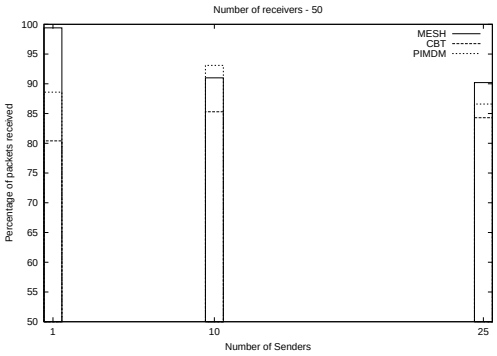

Figure 4: Percentage packet delivery (R-Receivers, S-Senders)

We compared the performance of MAP with the performance of PIM-DM and CBT using $ns-2$ [22]. These two protocols were chosen as representatives of tree-based multicast routing. We used the Internetwork Topology Models [19] [20] [21] from Georgia Tech to generate the topologies used in our experiments. The topologies created are flat random graphs that distribute vertices at random locations in a given place. Edges are added between pairs of vertices with probability α ($0 < \alpha \leq 1$). All the links(edges) are duplex and symmetric, i.e., links(i,j) and (j,i) have the same delay where i and j are nodes in the graph. The average node-degree of a topology is $2m/n$, where m is the number of links and n is the number of nodes.

We ran our simulations on a 100-node topology of average node degree three, which results in topologies that are not very densely connected. The performance criteria under study were the number of control packets required to maintain a group mesh or tree in the presence of link failures and the percentage of data packets delivered in different scenarios of simulation. The various scenarios were created by changing the number of receivers and sources, subjecting the network of 100 nodes to random link failures. All the simulations ran for over 1000 seconds. The prune reset timer for PIM-DM was set to 60 seconds.

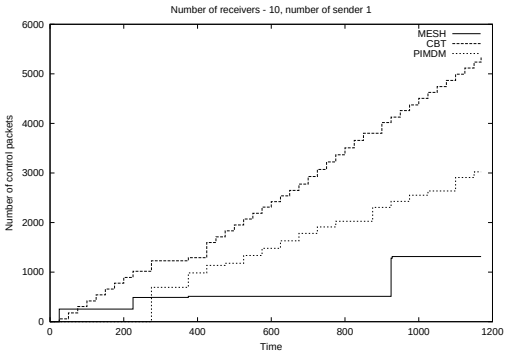

Figure 5: Control Packet Overhead 10 receivers 1 sender

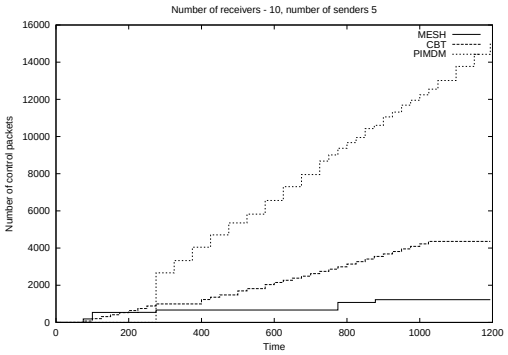

Figure 6: Control Packet Overhead 10 receivers 5 senders

For the experiments, links were failed at random every 25 seconds and then brought up after 100 secs. The scenarios used were 10 receivers and 1, 5 or 10 sources, and 50 receivers and 1, 10 or 25 sources.

Fig. 3 and Fig. 4 show the packet delivery rate for MAP, PIM-DM and CBT when links fail randomly. It is clear that MAP achieves better or similar packet delivery than PIM-DM and CBT. On the other hand, the overhead incurred by MAP is far less than PIM-DM and CBT. Figs. 5 to 9 show the signaling overhead incurred in building and maintaining the respective multicast meshes for MAP, and multicast trees for CBT and PIM-DM, in the presence of random link failures for various scenarios with different multicast members. The signaling overhead graphs for the three protocols

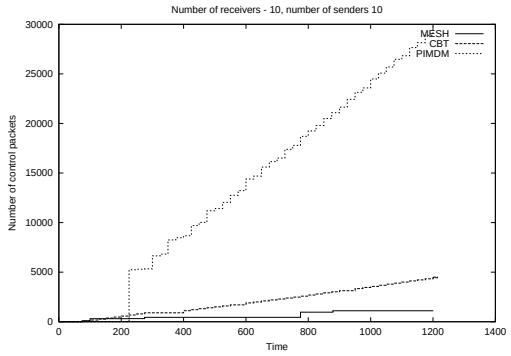

Figure 7: Control Packet Overhead 10 receivers 10 senders

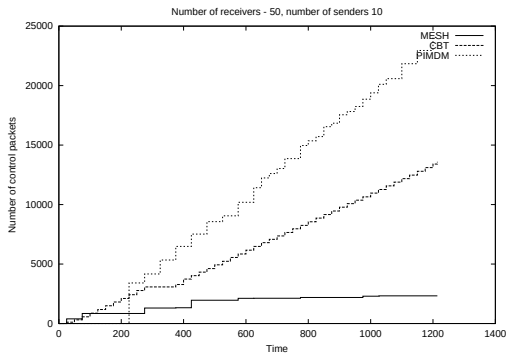

Figure 8: Control Packet Overhead 50 receivers 10 senders

show the cumulative number of control packets sent over time. It is clear that link failures hardly result in any control packet overhead in MAP. The results show that CBT incurs more overhead than PIM-DM in single-source groups, because it requires flushing and rebuilding subtrees and PIM builds a single source tree. The overhead of PIM-DM is more than that of CBT for the case of multiple sources, because PIM-DM builds multiple trees.

CBT was found to incur substantial looping of data packets in all scenarios, which is a problem identified previously [18]. To cope with CBT looping, data packets in the CBT experiments were dropped after they traversed more than 14 hops for all experiments other than the case of 50 members with 25 senders, in which case data packets had to be dropped as soon as they looped. The relatively poor packet delivery ratios found in CBT compared to those of MAP and PIM-DM are a direct consequence of looping. No looping was found in MAP and PIM-DM.

A key benefit of mesh-based multicasting is the ability to avoid looping of multicast data packets with very simple packet-forwarding mechanisms, rather than relying on flooding or complex tree-repair schemes capable of maintaining instantaneous loop freedom while multicast routing trees are modified, which are the approaches needed for PIM-DM or multicast routing based on shared trees [16, 18].

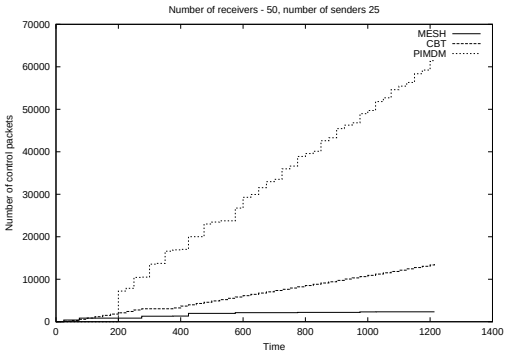

Figure 9: Control Packet Overhead 50 receivers 25 senders

5. CONCLUSIONS

The main contribution of this paper is the introduction of a new architecture for multicasting based on the construction of connected meshes rather than multicast routing trees. Meshes are more survivable than trees, because of path redundancy, and the distributed algorithms and mechanisms used to build meshes are simpler than those needed to build trees, because there is no need to guard against looping of packets at the time the routing structure is built or modified due to changes in topology or group constituency. As our simulation results exemplify, the signaling overhead of constructing multicast meshes is far smaller than the overhead of building and maintaining multicast trees, while more packets can be delivered in the presence of faults.

6. REFERENCES

[1] M. Ammar, G.C. Polyzos, and S.K. Tripathi (Guest Eds.), *IEEE Journal on Selected Areas in Communications*, Special Issue on Network Support for Multipoint Communications, Vol. 15, No. 3, April 1997.

[2] A. Ballardie, P. Francis, and J. Crowcroft, "Core Based Trees (CBT): An Architecture for Scalable Inter-Domain Multicast Routing," *Proc. ACM SIGCOMM'93*, pages 85–95, October 1993.

[3] B. Cain, S. Deering, and A. Thyagarajan, "Internet Group Management Protocol, version 3," Internet Draft, February 1999.

[4] C. Chiang and M. Gerla, "On-Demand Multicast in Mobile Wireless Networks," *Proc. IEEE ICNP 98*, Austin, Texas, October 14-16, 1998.

[5] S. Deering et al., "The PIM Architecture for Wide-Area Multicast Routing," *IEEE/ACM Trans. Networking*, Vol. 4, No. 2, April 1996, pp. 153–162.

[6] S. Deering, "Host extensions for IP multicasting," RFC-1112, August 1989.

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

[8] S. Deering et al. "An Architecture for Wide-Area Multicast Routing," *Proc. ACM SIGCOMM'94*, Cambridge, MA, 1994.

[9] S. E. Deering, "Multicast Routing in Internetworks and Extended LANS," *Proc. ACM SIGCOMM'88*, Stanford, CA 1988.

[10] W. Fenner, "Internet Group Management Protocol, Version 2," RFC 2236, November 1997.

[11] P. Francis, "Yoid: Extending The Intenet Multicast Architecture," Unrefereed Report, 2 April 2000.

http://www.isi.edu/div7/yoid/docs/index.html

[12] J.J. Garcia-Luna-Aceves and E. Madruga, "The Core Assisted Mesh Protocol," *IEEE Journal on Selected Areas in Communications,* Special Issue on Ad-Hoc Networks, Vol. 17, No. 8, August 1999, pp. 1380-1394.

[13] M. Gerla, et al., "On-Demand Multicast Routing Protocol," Internet Draft, http://www.ietf.org/internet-drafts/draft-ietf-manet-odmrp-00.txt, November, 1998.

[14] S. Kumar et al., "The MASC/BGMP Architecture for Inter-Domain Multicast Routing," *Proc. ACM SIGCOMM 98*, Vancouber, Briish Columbia, Canada, September 2–4, 1998.

[15] S. Paul, *Multicasting on The Internet and Its Applications*, Kluwer Academic Pub., 1998.

[16] M. Parsa and J.J. Garcia-Luna-Aceves, "A Protocol for Scalable Loop-Free Multicast Routing," *IEEE Journal on Selected Areas in Communications*, Vol. 15, No. 3, April 1997, pp. 316-331.

[17] R. Perlman et al., "Simple Multicast: A Design for Simple, Low Overhead Multicast," Internet Draft, February 1999.

[18] C. Shields and J.J. Garcia-Luna-Aceves, "The Ordered Core-Based Tree Protocol," *Proc. IEEE INFOCOM '97*, Kobe, Japan, April 1997.

[19] E. W. Zegura, K. Calvert and S. Bhattacharjee. *How to Model an Internetwork.* Proceedings of IEEE Infocom '96, San Francisco, CA.

[20] K. Calvert, M. Doar and E. W. Zegura. *Modeling Internet Topology.* IEEE Communications Magazine, June 1997.

[21] E. W. Zegura, K. Calvert and M. J. Donahoo. *A Quantitative Comparison of Graph-based Models for Internet Topology.* Accepted for publication in IEEE/ACM Transactions on Networking, December 1997.

[22] *The Network Simulator - ns-2*, http://www.isi.edu/nsnam/ns/, May, 2000.