A Tool for Debugging Internet Multicast

Deborah Agarwal and Sally Floyd*
Lawrence Berkeley Laboratory
University of California
Deb@nu.ece.ucsb.edu, floyd@ee.lbl.gov

Abstract

In this paper we describe a debugging tool that is an effective means of analyzing problems with multicast packet routing in a network. Multicast packet routing is a source-driven distributed calculation performed by the routers in a multicast network. The routes taken by multicast packets are difficult to predict manually due to the large number of variables that must be considered. The multicast route debugging tool allows off-line investigation of the route taken by a multicast packet and the effects of network modifications on that route. The tool has already proved useful in debugging the problems that have occurred in the experimental Internet Multicast Backbone. The multicast route debugging tool currently predicts multicast routes of packets using the distance-vector truncated-broadcast algorithm implemented for Internet multicast traffic. We will be upgrading the tool to allow the user to choose other multicast routing algorithms. [1]

1 Introduction

The ability to send multicast packets has recently been added to the Internet as an experiment, and several tools are already available to utilize this capability. Teleconferencing tools which provide the ability to multicast the audio and video of a conference are one example. There is also a multiparty whiteboard which allows multiple users to hold a meeting and interact on a virtual whiteboard in conjunction with voice communication. The whiteboard can be combined with the teleconferencing tools to allow the speaker to write on a whiteboard or to display overheads. These tools can be used for applications such as teaching, weather analysis, global mapping, medical image dissemination, and research collaboration to name just a few.

Until recently, teleconferencing tools required reserved communication channels and special equipment. This form of teleconferencing forces the user to go to the location containing the equipment. Recent experiments on the Internet have shown that it is possible to provide teleconferencing tools on an ordinary Unix workstation.

To support a teleconferencing tool, the communication network must provide N-way communication between the N participants of the conference. Each packet sent within the conference must be disseminated by the network to all of the sites in the conference. N-way communication is most easily achieved by sending multicast packets. A multicast packet is disseminated along the branches of a tree that spans the participants of the teleconference. Packets are duplicated by the routers at each branch of the tree.

If multicast capabilities were implemented using unicast (one-to-one) packets, a copy of each packet would need to be sent to each member of the conference. These packets would travel redundant links and waste network bandwidth. The other difficulty with unicasting the packets is the need to know the names of all of the participants of the multicast. Unicast routing is destination driven and each packet must be sent to a particular destination.

A multicast packet is instead addressed to a *group* of processors; in this paper, the group refers to the participants of a particular multicast session. The tree along which a multicast packet is routed depends upon the source of the packet and the locations of the members of the group. The sender of the packet need not determine the group membership before sending the packet. Each router in the multicast network decides whether to forward a packet based on local information and on information contained in the packet; the information used depends on the multicast routing algorithm run by the router. This information can include the packet source, the multicast group, the distance of the router from the source, and the distances of the neighboring routers.

*This work was supported by the Director, Office of Energy Research, Scientific Computing Staff, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

[1]

The recent Internet Engineering Task Force meetings have been multicast in the Internet [3]. This allowed people who were unable to attend the meetings to participate in them from their own offices. Also, President Clinton and Vice-President Gore have held several teleconferences that were broadcast over the Internet.

Since multicast capabilities are fairly new to the Internet, debugging tools have not yet been developed. Although there are unicast debugging tools, the multicast and unicast routing paradigms are completely different; multicast routing is primarily source driven whereas unicast routing is destination driven. Additionally, due to the experimental nature of the Internet multicast facilities, only a subset of the Internet routers can route multicast packets. This often means that the multicast and unicast packets follow different paths through the network despite having the same source and destination. The subset of the Internet currently providing multicast capabilities is referred to as the Multicast Backbone ($M - Bone$).

This paper describes a multicast route debugging tool (*mrdebug*) that has been developed for use in predicting the routes that multicast packets will take through a network. The tool also allows investigation of the effects of network modifications on multicast routes. The remainder of this paper is organized as follows; related work is discussed in Section 2, the multicast routing protocols are described in Section 3, the multicast route debugging tool is described in Section 4, an example of using the tool is presented in Section 5, how to obtain the multicast software is described in Section 6, and conclusions and future directions are covered in Section 7.

2 Related work

The only route debugging tools currently available on the Internet are intended for use in checking unicast routes. *Traceroute* [4] determines the current route used by packets unicast by a source to a particular destination. It sends packets from the invoking site toward the destination. Each packet is sent one hop farther toward the destination than the previous packet and causes an ICMP packet that identifies the last gateway reached to be sent back to the site running traceroute. Traceroute produces as output a list of the incoming interfaces along the route between the source and the destination. If N is the number of hops in the route, traceroute requires at least two times N packets to determine a route. This form of route debugging can be self-defeating when the network is congested since it adds to the network load.

If users want to determine why they are unable to receive multicast packets from another source, traceroute is of minimal use because routes in the network are not necessarily symmetric, and traceroute can only investigate the path from the source to a destination. Traceroute is also unable to predict multicast routes because, as mentioned earlier, the unicast route is not necessarily the same as the multicast route; not all routers in the network are multicast, and the algorithms used to determine the multicast and unicast routes are different.

One possible means of tracing multicast routing problems would be to create a multicast version of traceroute. The tool would print the current multicast packet routing tree for a particular source. Such a tool would put a significant load on the network since it would require at least $2N$ packets, where N is the number of nodes in the multicast tree. The load can be limited by implementing the multicast traceroute to only investigate a single branch at a time. The multicast traceroute tool would, however, be of limited use since users would be able to investigate only the paths of their own packets and users are more often concerned with problems related to receiving packets from other sources. To investigate the path from a source other than the user's site, the tool would have to send a packet to the source asking for a traceroute to be run and the results then transmitted back.

The multicast traceroute method itself is also limited since it provides no direct information regarding why a problem occurred. After receiving the trace of the multicast tree from a source, users who are not in the tree would still need some means of determining why they are not in the tree. This would involve additional queries in the network. A tool that could be provided for on-line debugging is a destination to source traceroute where the multicast path from the source to the destination could be investigated starting at the destination. This tool used in combination with a source traceroute could help in finding routing problems in the network.

Another unicast debugging tool, *ping* [4], allows users to determine if an interface is alive and reachable. This tool is used in conjunction with traceroute to investigate unicast routes, but is ineffective for multicast route debugging without a means of predicting the multicast packet routes.

The multicast route debugging tool (mrdebug) described in this paper has been designed as an off-line tool to predict the paths taken by packets multicast in a network while avoiding the network loading effects of on-line debugging tools. Mrdebug reads the description of the multicast network from a file and then predicts the multicast tree of any given source. This allows the user to investigate a multicast tree or path without loading down the network and potentially causing additional correlated failures. Mrdebug also allows investigation of the effects of network topol-

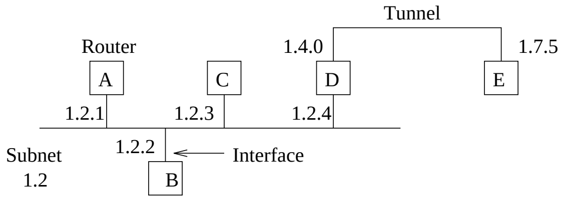

Figure 1: An example of a piece of a multicast internet. Routers A, B, C, and D have interfaces on subnet 1.2 and routers D and E are connected by a tunnel.

ogy modifications on the multicast routes.

3 Multicast on the Internet

A sample piece of a multicast internet is shown in Figure 1. In a multicast internet, different networks are connected at routers. A router can be connected to one or more networks and is responsible for determining the appropriate forwarding routes for packets it receives. Each router's connection to a network is referred to as an *interface*. An interface is an individually addressable entity that can send and receive packets. A network is termed a *subnet* if it provides complete connectivity between all the interfaces connected to it. An Ethernet or an FDDI network is an example of a subnet. Subnets generally have more than one interface attached to them and each packet broadcast on the subnet is received by all of the attached interfaces.

Tunnels can be used to provide point-to-point communication between any two routers that do not share a subnet. A *tunnel* forms a unicast path or datapipe through intermediate nonmulticast routers to the next multicast router.

3.1 The Multicast Backbone

The Multicast Backbone (M-Bone) consists of the Internet routers that have been reconfigured to route multicast packets in addition to their normal unicast traffic. Since the multicast packet routing algorithms [8, 10] are not yet part of the standard Internet Protocol suite, routers are normally reconfigured to route multicast packets only if there is sufficient user demand. Because only a subset of the routers in the Internet currently provide multicast capability, the M-Bone uses tunnels to connect multicast routers.

Each multicast router's subnet interface and tunnel is configured with a metric and threshold. The *metric* indicates the cost of sending a packet out the interface or tunnel. The *threshold* indicates a minimum time-to-live that must be contained in the multicast packet for it to be forwarded out the interface or tunnel.

The scope of a multicast packet can be limited to a local area by appropriately setting the $time-to-live$ field in the packet. A packet is forwarded out of an interface only if the time-to-live of the packet is greater than one and is greater than or equal to the interface threshold. Each time a packet is forwarded, the time-to-live in the packet is decremented by one. For example, a packet broadcast with a time-to-live of two will be broadcast out of only those interfaces that have a threshold of one or two and the packet will not be forwarded. The scope of a multicast packet can be limited to a local area by setting the thresholds on interfaces that lead out of the area to an appropriately high value and making sure that the packet is broadcast with a time-to-live that is less than this threshold.

At the moment, interfaces and tunnels are configured manually by a system administrator and the metric and threshold are chosen heuristicly based on ad hoc guidelines. If a router's metric and threshold attributes are incorrectly configured, the router can create havoc in the M-Bone by accepting more packets than it can handle or by accepting packets but refusing to forward them.

3.2 The distance-vector truncated-broadcast algorithm

Several algorithms have been proposed for use in routing multicast packets in the Internet. The distance-vector truncated-broadcast algorithm developed by Deering in [9] is currently used by the routers on the M-Bone to route multicast packets. This algorithm is a refinement of the reverse-path forwarding algorithm proposed by Dalal and Metcalfe [6]. The result of the algorithm, which is explained later, is that a packet is disseminated down a minimum spanning tree from the packet's source. Other algorithms that have been proposed are the link-state multicast routing algorithm [9] and core-based trees [1].

Although the distance-vector truncated-broadcast algorithm is currently implemented in the M-Bone, multicasting has not been adopted as part of the Internet standard. The distance-vector truncated-broadcast algorithm is described here because it is the algorithm used on the M-Bone and, therefore, by the mrdebug tool. Currently, M-Bone routers maintain multicast routing tables completely separately from the unicast Internet routing tables. This is due primarily to the experimental nature of the M-Bone, and it is expected that the routing tables will eventually be merged and multicast capabilities integrated into the Internet Protocol.

In distance-vector truncated-broadcast routing, the multicast routers use a distance-vector routing algorithm to determine their minimum distance and next-hop router to each subnet in the network. The distance-vector routing algorithm used in the M-Bone is actually the Bellman-Ford algorithm [2]. Distance vector information is maintained at a multicast router by exchanging distance tables with neighboring multicast routers. When a multicast router exchanges a distance table with a multicast neighbor, it reports its distance to each of the known destinations in the multicast network except those whose next hop is the subnet or tunnel the table will be sent on; those entries are reported as being infinite distance. When a multicast router receives a distance table from a neighbor, it adds its own distance to the neighbor to the entries of the distance table reported by the neighbor. The router then updates its own table by replacing any of its entries that contain a larger metric than the one contained in the neighbor's adjusted distance table; the table is not updated if the metrics are equal. The router also updates the next-hop to be the neighbor whose update changed the table entry.

Details of the algorithm executed by the router in distance-vector truncated-broadcast will be given here to aid in later describing the mrdebug tool. When a multicast packet is forwarded from a neighboring router, the packet is accepted only if the neighbor is the next-hop router to the source of the packet. This means that the packet has arrived on the shortest path from the router to the source. The packet, if accepted, must be copied to the interfaces with neighbors that consider this router their next hop to the source of the packet. The router maintains data structures for each source termed *children* and *leaves* to determine the interfaces to copy packets onto. A *child interface* has neighbors that have either a greater distance to the packet source or an equal distance and higher address. A *leaf interface* is a child interface with no neighboring routers reporting the source as infinite distance. This means that there are no routers on the subnet that consider this router to be their next-hop router to the source of the packet.

A packet, if accepted, is copied by the multicast router to the interfaces that are children but not leaves for the given source. A packet is copied to a leaf interface if there are hosts on the subnet that are interested in that packet. A packet is never copied to a leaf interface that is a tunnel, because there are only routers on the ends of a tunnel. Each time the distance-vector routing table is updated, the router also updates its children and leaves.

The time-to-live field provides the only current means of limiting scope of a multicast packet in the M-Bone. Multicast packets sent to a multicast session with widely spread participants will need to be sent with a large time-to-live and will be received by most of the M-Bone. The multicast routing algorithms will be modified soon so that sites involved in a multicast session will join a group and packets will be multicast only within the group. The modifications required in the distance-vector truncated-broadcast algorithm to handle groups are described in [9] and involve the routers pruning the multicast tree to eliminate branches that do not contain members of the group. The time-to-live field will still be used to limit scope in conjunction with the multicast pruning mechanism.

The ability to prune the multicast tree will reduce unnecessary bandwidth use by groups that are not confined to a local area but have only a few members. Further modifications to the multicast protocols for sparse groups are still under investigation. Some proposed algorithms for handling sparse groups are detailed in [1, 7].

4 The multicast route debugging tool

Packets in a multicast network are disseminated across a wide area and travel through many routers. Each router makes the decision individually whether to accept an incoming packet and which neighboring routers should receive copies of the packet. These routers base their decisions on their local interfaces, their distance to their neighbors, and information provided by their neighbors. The distributed nature of the routing calculation makes debugging difficult. When participating in a multicast group, users may find that they are unable to receive packets from some subset of the participants and that some other subset is unable to receive their packets, a frustrating experience.

Many factors can lead to a portion of the group being unable to receive packets from a particular source. The first possibility is that there may not be any paths through the network that connect the source and the destination due to failure of a primary network link or router. It may also be the case that an insufficient time-to-live was specified in the packet such that some router on the path discarded the packet. Sometimes two sites that are geographically near use circuitous routes to multicast to each other due to the configuration of the network. This can lead to excessive packet loss and can be corrected with minor changes to the network configuration.

The mrdebug tool was developed to allow a user to investigate the multicast routes used by packets from any given source and to determine the effects of a specified time-to-live in a packet. This allows a user to determine whether the packet is being limited to the desired area. The user can also use mrdebug to investigate changes to the network topology and to observe the results before modifying the actual network topology.

4.1 Network description

Mrdebug is an off-line tool that takes network and subnet description files as input. An automated tool for generating the network and subnet description files has been built and its output is available via ftp (see Section 6). The *network description file* contains a description of the topology of the multicast network, and the *subnet description file* contains a list of the subnets in the multicast network. Each subnet has a mask associated with it and all interfaces on the subnet will have the same masked address. Mrdebug reads in the subnet description file to determine the subnets and their associated masks. It then reads the network description file. Each interface's subnet is determined by masking the interface's address. If the masked address matches a subnet address, then the interface is considered to be on that subnet.

The network description is stored in a graph adjacency list with each router and each subnet representing a node in the graph. The edges of the graph are the tunnels and subnet connections. The subnets are modeled as separate nodes to allow easy simulation of the effect of sending a broadcast packet on a subnet. Connections to a subnet or tunnel from a router are given the metric and threshold used in the actual network, as specified in the network description file.

4.2 Prediction of multicast routes

This section describes the methods used by mrdebug to predict multicast routes. The distributed operation of the multicast routing protocol is simulated by mrdebug as a single process. Mrdebug uses Dijkstra's algorithm [5] to calculate the distance-vector routing table entries. Ties are resolved by using the lower address neighbor as the next hop. Distances between routers are determined by using the metric of the edge in the reverse direction, as described in [9]. Each router's distance-vector routing table entry in mrdebug lists the next-hop neighbor and the minimum distance to the specified source. The next-hop neighbor is expected to forward packets from the source to this router.

Mrdebug calculates whether an interface should receive a copy of packets from a given source by looking directly at the neighboring router's distance vector routing table and using the algorithms specified in [9]. A router may receive a redundant copy of a message if a subnet it is attached to is used by another router to receive messages along the multicast tree. A router may also receive redundant copies of a message if a message is forwarded through a tunnel that is not the router's minimum distance path to the source. This behavior is predicted in [9]. When a redundant forwarding operation is found, mrdebug prints the interfaces at the receiving router with a metric of -1 on the redundant branch of the tree. These interfaces will also appear in the tree along their shortest path with their proper metric.

The mrdebug tool uses by default the distance-vector truncated-broadcast algorithm to predict multicast routes and does not implement pruning for multicast groups. The pruned tree of a source multicast to a group is always a subset of the source's full multicast tree so the ability to show the pruned tree would be of limited additional value. Also, the multicast groups are expected to be relatively dynamic making it difficult for an off-line tool to predict the current pruned tree for the multicast group. Mrdebug can also display multicast routes determined using link-state routing, and has been built to allow the future addition of other multicast routing algorithms.

4.3 Configuration

Mrdebug was written in C and uses a textual interface. This allows users access to the tool from any type of terminal. There are plans to provide a graphical representation of the tree and path outputs in X windows, but this has not yet been implemented. The input files defining the network can be generated manually by the user. Automatically generated files describing the M-Bone and the mrdebug tool are available via *ftp*, see Section 6 for details.

4.4 User interface

Mrdebug can be run interactively allowing the user to investigate source multicast trees, multicast paths to destinations, or the effects of changes made to the network by the user. At any time during operation of the tool, the user can specify or respecify the interface to use as the source of the multicast or the destination of the multicast. The user is also able to specify an output file for all program output to allow later analysis and comparison.

Any of several displays can be requested by the user. When invoking the multicast tree option, the user can specify a time-to-live for packets from the source and view the resulting truncated multicast tree (see Section 5 for examples). The multicast trees are printed textually with indentation used to show tree level. Mrdebug also allows the user to display the path through the multicast tree to a single destination.

A problem in predicting multicast routes that use a distance-vector routing algorithm is the random behavior when equal-length routes exist between a source and a destination. If a change in the network topology occurs in the actual network, the multicast routing protocol will change from an existing path to a new path only if the new path is of shorter distance. Thus, the tree printout produced by mrdebug may not exactly match the actual multicast tree if there are multiple paths of equal distance from the source to some of the destinations. The path printout will, however, enumerate all of the possible paths that could be in use by the multicast routing protocol to reach a destination. The random behavior of the multicast routing algorithm in this situation will be made deterministic in the next release of the multicast code for the M-Bone. In the meantime, the current path in use by the actual multicast routers can only be determined by an on-line tool.

When a multicast tree is printed out, mrdebug can display a list of the network interfaces that are unreachable. This includes interfaces that were not reachable due to the specification of a time-to-live for the packets. The list of unreachable interfaces can be useful for debugging since sites that are having trouble receiving a multicast should appear on this list. If a limiting time-to-live was not specified with the multicast tree, interfaces are unreachable only if they are down or in a disconnected subgraph of the network.

The mrdebug user can interactively make network topology changes and redisplay the multicast tree and unreachable interfaces. The network topology can be changed by removing an interface or tunnel, by adding an interface or tunnel, or by changing the metric and threshold of an interface or tunnel. This capability can be used to investigate new topologies before implementation or to debug existing problems.

The mrdebug tool can also be run in batch mode to provide input to graphical interface or multicast route analysis tools.

5 Example

To better illustrate the need for, and usefulness of, a multicast route debugging tool, we now present a small example. The network shown in Figure 2 consists of ten routers, three subnets and three tunnels. The routers are represented in the figure by black circles and the subnets are represented by horizontal lines. Each arrow that goes directly from one router to the next represents a single direction of a tunnel. The numbers at the routers are the interface addresses associated with the router. When there is more than one interface at a router, each interface is listed near the connection representing that interface. The annotations along the edges in the network give the metric/threshold for the edge.

A request for the multicast tree rooted at the interface 128.1.0.0 in Figure 2 would produce the following output from mrdebug:

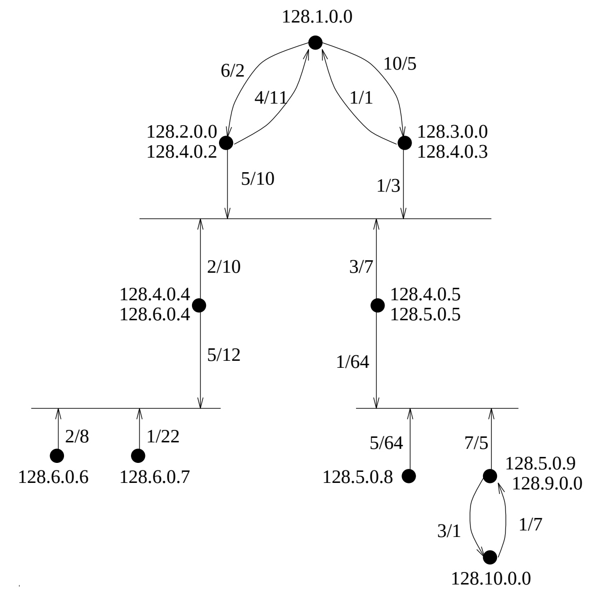

Figure 2: A sample multicast network. The routers are represented by dots, the subnets are horizontal lines and the tunnels are edges directly connecting two routers. Each edge is annotated with its metric/threshold.

```
128.1.0.0, 0/0/0
  128.3.0.0, 1/1/5
    128.4.0.3, 2/1/5
      128.4.0.4, 3/3/5
        128.6.0.4, 4/3/5
          128.6.0.6, 5/5/14
          128.6.0.7, 5/4/14
      128.4.0.5, 3/4/5
        128.5.0.5, 4/4/5
          128.5.0.9, 5/11/66
            128.9.0.0, 6/11/66
              128.10.0.0, 7/12/66
          128.5.0.8, 5/9/66
      128.4.0.2, 3/−1/5
        128.2.0.0, 4/−1/5
  128.2.0.0, 1/4/2
    128.4.0.2, 2/4/2
```

The numbers printed after each interface name are tree level/metric-to-here/time-to-live required to be in a packet to reach here. The indentation shows the tree level. Each interface is printed in the tree directly below the interface at the next higher level of the tree and indented one position. If a packet departs from a router through a different interface than it arrives on, the departure interface is shown indented one position below the arrival interface.

In the above tree printout, interface 128.3.0.0 receives packets multicast by 128.1.0.0 directly from the tunnel interconnecting them. The metric used to determine the cost of using this edge for the multicast is 1, the metric of the edge in the reverse direction. The threshold of 5 on the forward direction of the tunnel determines the time-to-live required by a packet to reach 128.3.0.0 from 128.1.0.0. Entries further down this branch of the tree will be printed with a time-to-live no less than 5.

The packet is received from interface 128.3.0.0 by interface 128.4.0.3 which is located at the same router; this is indicated by no increase in the metric. The packet is then broadcast on subnet 128.4 by interface 128.4.0.3. Interface 128.4.0.3 has an outgoing threshold of 3, so packets from 128.1.0.0 need a time-to-live of at least 4 to be sent out this interface. Because this is less than 5, this does not add any new restrictions. The packet is next received from the subnet by interfaces 128.4.0.2, 128.4.0.4, and 128.4.0.5. These interfaces are all at the same tree level so they are each printed at the same indentation level below 128.4.0.3. Interfaces 128.4.0.2 and 128.2.0.0 are printed with metrics of -1 because, although they received the packet from 128.4.0.3, it was not from the next-hop router to the source. These interfaces were expecting the packet to come directly from 128.1.0.0. This is indicated by their appearance at their proper place in the tree at the bottom of the printout.

If the user requests the multicast tree for a source interface such as 128.4.0.11, which is not a multicast router but is on a subnet, the mrdebug tool will print the tree assuming the packet originated directly on the subnet. This is because the interface threshold for 128.4.0.11 is not known. The multicast tree for packets from 128.4.0.11 would look like this:

```
128.4, 0/0/0
    128.4.0.4, 1/2/0
        128.6.0.4, 2/2/0
            128.6.0.6, 3/4/12
            128.6.0.7, 3/3/12
    128.4.0.5, 1/3/0
        128.5.0.5, 2/3/0
            128.5.0.9, 3/10/64
                128.9.0.0, 4/10/64
                    128.10.0.0, 5/11/64
            128.5.0.8, 3/8/64
    128.4.0.2, 1/5/0
        128.2.0.0, 2/5/0
            128.1.0.0, 3/11/11
    128.4.0.3, 1/1/0
        128.3.0.0, 2/1/0
```

The thresholds of zero indicate that the packet, since it was assumed to originate on the subnet, did not require a time-to-live to be specified in the packet in order to reach the interfaces on the subnet. If we look only at the multicast path from subnet 128.4 to interface 128.1.0.0 the printout from mrdebug will show

```
128.4, 0/0/0
    128.4.0.3, 1/1/0
        128.3.0.0, 2/1/0
            128.1.0.0, 3/11/2
```

and

```
128.4, 0/0/0
    128.4.0.2, 1/5/0
        128.2.0.0, 2/5/0
            128.1.0.0, 3/11/12
```

There are actually two paths with lengths equal to the minimum distance between subnet 128.4 and interface 128.1.0.0: one through interface 128.4.0.3 and one through interface 128.4.0.2. Since the actual multicast algorithm may be using either path, both are printed for the user. The alternative paths are not printed in the multicast tree because it would make the tree difficult to understand.

The mrdebug tool provides many ways to debug network problems. One way is to input a map of the fully operational network and print the path to the destination which is not receiving the multicast. The user can then ping the interfaces along the path to identify the one that is down. Alternatively, the current network state can be input to the mrdebug program and methods to reconnect the multicast tree to the unreachable interfaces can be investigated. Another use of the mrdebug tool is to aid in determining an appropriate time-to-live for a group to ensure that a multicast conversation remains confined to the desired local area. The results of such an investigation may indicate that the thresholds on some of the interfaces of routers in the local area have been set inappropriately.

6 Software availability

All of the software described in this paper is available via anonymous *ftp* to various sites on the Internet. Readers wishing more information regarding the M-Bone should retrieve the file mbone/faq.txt from venera.isi.edu (128.9.0.32). The faq provides information about how to get connected to the M-Bone, obtaining kernel modifications to support multicasting, supported hardware platforms and where to find the current multicast applications. A map of the current M-Bone is available from parcftp.xerox.com (13.1.64.94) in file pub/net-research/mbone-map*. The mrdebug program and automatically generated input files can be obtained from ftp.ee.lbl.gov (128.3.112.20) in mrdebug.tar.Z.

7 Conclusion and future directions

Debugging of multicast routing is a fundamentally different problem than debugging of unicast routing and requires a new set of debugging tools. A primary difference is that multicast packet routes are source rather than destination driven. In addition the current M-Bone is experimental and is composed of only a subset of the Internet so many routes through the Internet are not available to multicast packets. For the above reasons unicast route debugging tools are of limited use in debugging multicast routes.

The multicast route debugging tool (mrdebug) allows users to determine the routes taken by multicast packets in a network. The users can print the multicast routing tree of a source or the multicast path from a source to a destination. Because it runs off-line, mrdebug does not contribute to network loading. It can be used to debug problems as they occur or to investigate the effects of changes to the network prior to making the changes in the actual network.

A potential problem that must be dealt with in the mrdebug tool is scaling. The base mrdebug tool will not need to change if the M-Bone changes to hierarchical routing since each domain would be routed separately. A separate instance of the tool can be run for each domain. The primary scaling problem will be in gathering topology. The current M-Bone contains approximately 650 nodes and collection of the topology takes over an hour. The entire Internet contains approximately 14,700 nodes so a complete topology update would take about a day. This problem has not yet been addressed but can easily be solved through the use of incremental topology updates.

Mrdebug is currently configured to use distance-vector truncated-broadcast routing to predict the routes taken by multicast packets in the network. The mrdebug tool has been built to allow easy incorporation of new multicast routing algorithms as the need arises, and link-state routing has already been incorporated into mrdebug. We will continue to upgrade and refine the mrdebug tool to keep pace with developments in multicast routing.

The mrdebug tool is well suited to network design and many debugging tasks but does not eliminate the need for an on-line tool. An on-line tool is needed to provide information regarding dynamic behaviors in the network such as transient failures and multicast groups.

Mrdebug has already proven useful in debugging the experimental multicast routing in the Internet; problems encountered during setup of the most recent Internet Engineering Task Force meeting videocast were diagnosed using the mrdebug tool.

Acknowledgments

The multicast route debugging tool was built with the guidance and aid of Van Jacobson at the Lawrence Berkeley Laboratory.

References

[1] A. Ballardie, P. Francis, and J. Crowcroft. Core based trees (CBT) an architecture for scalable inter-domain multicast routing. In *ACM SIGCOMM Symposium on Communications Architectures and Protocols*, pages 85–95, September 1993.

[2] R. E. Bellman. *Dynamic Programming.* Princeton University Press, 1957.

[3] S. Casner and S. Deering. First IETF Internet audiocast. *ConneXions*, 6(6):10–17, June 1992.

[4] D. E. Comer. *Internetworking With TCP/IP Volume I: Principles, Protocols, and Architecture.* Prentice Hall, 1991. 2nd Edition.

[5] T. H. Corman, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, 1990.

[6] Y. K. Dalal and R. M. Metcalfe. Reverse path forwarding of broadcast packets. *Communications of the ACM*, 21(12):1040–1048, December 1978.

[7] S. Deering, D. Estrin, D. Farinacci, and V. Jacobson. IGMP router extensions for routing to sparse multicast-groups. available via ftp at cs.ucl.ac.uk:/darpa/idmr-archive.Z, July 1993.

[8] S. E. Deering. Host extensions for IP multicasting. RFC 1112, August 1989.

[9] S. E. Deering. *Multicast Routing in a Datagram Internetwork.* PhD thesis, Stanford University, December 1991.

[10] D. Waitzman, C. Partridge, and S. Deering. Distance vector multicast routing protocol. Internet RFC 1075, November 1988.