A Practical Multicast Monitoring Scheme

Joerg Walz Brian Neil Levine
jwalz@cs.umass.edu brian@cs.umass.edu
Department of Computer Science
University of Massachusetts
Amherst, MA 01003

Technical Report: 30 – 2000

June 22, 2000

Abstract

Deployment of multicast routing services in corporate networks and Internet Service Providers is still tentative. Among other problems, there is a lack of monitoring and management tools and systems. Previous work in multicast management has failed to address the scalability problem present in multicast fault isolation and reporting. We propose a hierarchical, passive monitoring scheme, HPMM, that relies on a series of pre-deployed, self-organized monitoring daemons. With HPMM, fault message aggregation and local fault detection and isolation is more efficient than previous approaches. HPMM satisfies a number of design goals: scalable reporting; fault isolation; no dependencies on multicast routing for reporting; no modifications to existing routing or diagnosis protocols; and ease of deployability. The tradeoff of using HPMM is it leverages a large number of software daemons deployed along a local domain. We compare the performance of HPMM and previous work with a simulation.

1 Introduction

Deployment of multicast routing services in corporate networks and Internet Service Providers is still tentative. Among other problems that have deterred multicast deployment, such as a viable commercial service model [18], there is a lack of monitoring and management tools and systems. Successful deployment of services requires that administrators are able to supervise correct function and to detect and isolate faults. However, supervising multicast traffic is a difficult problem. For unicast services, network management systems are able to track and monitor traffic flows in various ways. For multicast, each connection involves multiple hosts and is therefore more difficult to monitor. It is currently not possible to monitor all ongoing multicast transmissions, whether within one administrative domain or over multiple domains. In this paper, we offer a new scalable and practical multicast management tool called the Hierarchical Passive Multicast Monitor (HPMM) that does not suffer from many of the problems of previous solutions.

Several monitoring and fault detection tools are currently available to test multicast transmission in a network [20, 4, 30]. Unfortunately, each of these is only meant for a specific scenario — like RTPMon [8] for RTP-based multicast — or for a specific fault — like mtrace [22], which is able to track an incoming multicast flow back to its source for a specific receiver.

A more promising approach is offered by the Multicast Reachability Monitor (MRM) protocol [7], which monitors intradomain multicast flows by supervising test traffic. This requires the test traffic to be configured so that the paths of existing multicast traffic in the network is covered by the flow of test traffic. This is not an easy task, since knowledge about ongoing traffic flows has to be gathered. The problem is then to define monitoring tests is a way so that additional traffic is minimized, but nevertheless all flows are covered. A more serious problem is that MRM is not likely to perform well in large-scale environments because faults in multicast traffic are almost always correlated over multiple receivers, causing an implosion of reports at the central collection point. In this paper, we show that a single, centralized monitor station, as used with MRM, results in a bottleneck for multicast management reporting.

As an alternative approach, we introduce a distributed multicast monitoring scheme called Hierarchical Passive Multicast Monitor (HPMM), which relies on a series of pre-deployed monitoring daemons self-organized in a hierarchical fashion according to network topology and ongoing multicast traffic in the network. Accordingly, with HPMM, fault messages aggregation and intradomain fault detection and isolation is more efficiently provided. HPMM satisfies a number of design goals: scalable reporting; fault isolation; no dependencies on multicast routing for reporting; no modifications to existing routing or diagnosis protocols (such as IGMP mtrace); and ease of deployability. The tradeoff of using HPMM is it leverages a large number of software daemons deployed along a local domain. However, we believe this to be reasonable as such daemons could be co-located with SNMP installations, and furthermore, the task of deployment by network administrations occurs only once.

This paper is arranged as follows. In Section 2, we review previous work in multicast management. In Section 3 we define different faults that can be found in a network infrastructure attempting to support multicast. In Section 4 we define HPMM. Section 2.1 gives justification for the necessity of multicast monitoring. In Section 5, we present simulations that show HPMM is able to monitor all multicast traffic within an administrative domain without the implication of excessive additional traffic. Our comparisons show HPMM performs better than MRM and even recent work that requires modifications to multicast routing protocols [35]. We offer concluding remarks in Section 6.

2 Related Work

In this section, we discuss why multicast monitoring is different from unicast monitoring, and we review previous work in the area of multicast monitoring and debugging. Currently available tools either address specific debugging problems with multicast, such as route discovery, or imply considerable management overhead by introducing additional traffic to the network.

2.1 Justification

The development of multicast monitoring techinques is a necessary component for success large-scale multicast deployment. Even though the necessity of manageability of network traffic in general is commonly acknowledged, multicast management is inherently different from regular unicast and broadcast traffic management. For multicast to be offered as a successful commercial service, it must be offered at a consistently high quality [18]. Whenever faults occur, fast and accurate fault detection and isolation must be applied. Existing tools fail in this capacity [6].

Because faults are correlated among multicast receivers, monitoring can be difficult. Recent research [41, 9, 12, 29, 25, 3, 5, 11] has shown that current multicast applications in the Internet see considerable amount of spatially correlated faults, mostly packet loss — sometimes correlation can reach to 20% of the receivers [41], even across domains. Using legacy unicast management, like SNMP, it is difficult to find the fault correlation areas, since the management station has to differentiate between correlated and uncorrelated faults. Additionally, the management station wil be flooded with reports if a fault affects multiple hosts. This problem is not limited to SNMP. In fact, every management scheme based on a central station managing unaggregated reports faces this unscalability (see Section 5 for related simulation results). An scheme to reduce the number of fault reports without reducing fault response time is therefore needed for multicast monitoring.

2.2 Related Work

Monitoring approaches for multicast traffic have not been able to provide a complete suite for a network administrator to easily identify multicast faults in the network within one piece of software (except perhaps for the most recent addition to HP OpenView, called mmon [27]). Although a variety of tools are available, most of them do not interact with each other. This problem arises from the fact that initial multicast deployment has been restricted to experimental arenas administered by experts. (Almeroth provides an excellent overview of multicast evolution [2].) During this phase, debug-level tools were developed to verify specific problems in early multicast usage. Even though later on the functionality of these tools were expanded, they are still not intended to be used by a non-expert user.

The most used representative of this category is *mtrace* [22]. Developed by Fenner, mtrace discovers the reverse path of a multicast group from any given receiver back to the source using special diagnostic messages defined in IGMPv2 [21]. It displays all routers on the path, along with protocol and traffic statistics, such as packet loss and packet delay. Another generic tool is *mrinfo* [20], which returns the current status of a multicast capable router. Mrinfo has the same scalability problems as mtrace: diagnosis is limited to a single location. Several other tools are available that use similar approaches; a summary of these is given by Thaler [39].

The next generation of tools try to provide a better summary of network statistics. By combining multiple

basic tools, they are able to present a group-based overview of multicast services. Included in this class of tools are MView [26], RTPMon [8], and MHealth [30]. A comprehensive overview of these tools is provided by Almeroth [1]. These tools have the advantage of displaying network statistics and faults, notably packet loss and topology problems for the monitored group. Unfortunately, they suffer from scalability issues as they are not suited to monitor all multicast groups in the network. Additionally, some of are restricted to RTP-based multicast traffic.

The usage of the Real-time Transport Protocol (RTP) [36] and Real-time Transport Control Protocol (RTCP) implies several scalability problems. First, statistical reports are multicast to all receivers of the transmission. Second, RTCP increases the intervals between reports for large groups and cannot be adjusted by the monitoring tool.

Another approach to monitoring multicast services is to constantly query all entities in the network for statistics. This is done for intra-domain networks through the *Simple Network Management Protocol* (SNMP) [17], which relies on *Management Information Bases* (MIB) [33, 32] to provide local statistics. For inter-domain usage, proprietary software agents are used that run at each node and report to a central management station. The most popular representative of the former one is HP OpenView. Multicast extensions have only recently been added to its functionality with a prototype application called *mmon* [27]. For the second technique, the CAIDA taxonomy group [13] has developed a software agent that provides statistics to servers located at the CAIDA site.

2.3 Recent Work

The last category of multicast monitoring approaches were introduced through the development of the *Multicast Reachability Monitor* (MRM) protocol [6, 7]. MRM is the first approach to define a completely new protocol explicitly for multicast monitoring. We believe MRM is the most promising method of the tools described in this section and we place our proposal in direct comparison to MRM. It is therefore necessary to describe MRM in a more detailed fashion. (Readers are encouraged to consult the full protocol specification [7] for a definitive description of MRM.)

MRM can be deployed at hosts or routers, and defines two basic types of entities: the MRM manager and MRM testers. The manager acts as a controlling entity and is responsible for setup, maintenance and data collection from MRM testers. Testers can be configured to be either a test sender (TS) or test receiver (TR). The functionality of MRM includes the following:

1. Definition of multicast test senders and receivers. The MRM manager appoints TSs and TRs by sending unicast requests to each entity to participate in the test, along with specifications for the test, e.g., a multicast address to monitor.

2. Initiation of multicast transmissions by test senders to a specified multicast test address.

3. Monitoring of multicast test traffic at test receivers. Each receiver observes incoming multicast messages according to its configuration given during setup and sends status reports to the MRM manager. Reports can be sent either by unicast or by multicast depending on fault conditions.

4. Evaluation of status reports at the MRM manager.

Even though MRM provides several methods to deter flooding the MRM manager with fault reports, we show in Section 5 that MRM does still not scale well enough for large-scale deployment. Furthermore, MRM can be configured to use multicast itself to report faults. We find that this implies both the problem of lost reports during multicast outages and reduced fault detection ability. Nevertheless, we believe MRM is an important step in terms of multicast monitoring as it does not rely on unicast management protocols and attempts a complete solution. Section 2.1 discusses this point in detail.

The last approach we review here has been recently introduced by Reddy et al [35]. Their technique isolates faults using IGMP MTRACE messages initiated by receivers and then gathering statistics. So that every receiver trace does not implode at the multicast source, Reddy et al introduce a mechanism to reduce the trace length while still covering all paths in the multicast tree. Relying solely on mtrace for diagnosis, this approach is neither practical nor accurate. First, mtrace does not provide reliable statistics, and sometimes none at all. Second, network administrators commonly disable mtrace due to security considerations: network probing through mtrace and the resulted router discovery by users is not wanted. Third, the technique requires several modifications to multicast routing protocols to operate efficiently and therefore is not deployable in the near term. Finally, we show in Section 5 that even with router modifications, this scheme does not operate as efficiently as the approach we introduce in this paper which does not require such modifications (nor does the approach operate as efficiently as MRM). In addition to these practical issues, the approach is not proven to detect multiple simultaneous faults [35] and lacks considerations on how multiple groups can be handled efficiently.

3 Fault Definitions

In this section, we define types of faults that may occur in a multicast network. We refer to these definitions to prove the correctness of our monitoring approach. Only faults relevant for multicast are mentioned here; we rely on legacy unicast management systems to detect other problems. The following definitions owe much to previous work in this area [7, 39].

There are several more possibilities than those we list here that affect the correct functionality of multicast. We refer the reader to a complete summary given by Thaler and Aboda [39], including session announcement problems that are not detectable by our approach.

Topological disconnectivity. The main causes of network partitions are broken links, router crashes, route flapping or misconfiguration, or general topology changes. Usually, these faults result in a total loss of

connectivity both for multicast and unicast traffic. But since multicast and unicast don't always follow the same path, it may occur that part of a network is disconnected for multicast, but still reachable for unicast traffic. We define this case as partial disconnectivity.

Forwarding errors. This type of fault results from misconfiguration in multicast routing tables or incorrect function of the forwarding engine. The effect can be parts of the network receiving multicast transmissions incorrectly or not at all. It may occur that multicast packets are forwarded in the wrong direction and dropped due to reverse-path forwarding checks.

Packet loss. This fault type is more of an effect perceived at receivers than actual network faults; it is nevertheless useful to define a separate category for packet loss. Packet loss is often due to congestion in the network. Router queue are overflowed and therefore packets dropped. A network management system has to be able to detect excessive packet loss rates and isolate them in terms of location, affected area and, if possible, the cause.

Duplicates. In case of an incorrect entry in the forwarding table, routers incorrectly forward packets in directions that are already serviced in other ways. Los prune messages are possible cause of this fault.

From a user point-of-view, these problems are mostly transparent. In fact, a receiver participating in a multicast session is only able to distinguish correct and incorrect function by observing packet loss rate, packet delay and delay jitter. In worst case, it is also able to constitute that no multicast transmission is received at all.

4 HPMM

In this section, we define our proposal, called *Hierarchical Passive Multicast Monitoring* (HPMM), which monitors and detects faults in ongoing multicast traffic. HPMM is based on software monitors arranged hierarchically according to network topology and existing multicast routes. When a fault is detected, agents first contact their *upstream parent*, which is the monitoring agent that is closer to the source of the multicast group for which the fault been detected. The upstream parent is able to determine whether the fault is correlated. If so, it contacts its own upstream parent reporting the same fault. If not, the fault occurrence can be isolated as existing between the node that has perceived the fault and this node. This is recorded and reported to a management station. If the management station is unreachable, the report is archived until connectivity resumes for later forensics.

With this method, only a very limited number of messages for each fault correlation area are sent to the management station. The messages determine the part of the network the problem occurs in and the most likely problem type. Therefore, HPMM provides localized fault detection and isolation with scalable reporting.

Hierarchical protocol processing is a well-known technique. Several reliable multicast protocols [34] use this idea to handle scalable error recovery. Our approach is validated by observations on multicast packet

loss correlation [28]. A similar approach is offered by the *Globally Distributed Troubleshooting* (GDT) protocol, which also uses a tree-like structure of *expert location servers* [38]. GDT is used for automated troubleshooting in inter-domain areas and relies on *domain-specific expertise modules* for fault detection and isolation.

4.1 Basic functionality

HPMM requires nodes in the network to implement the HPMM functionality by running a software *agents*. The preferred node type for agents are the application-level systems located at routers, but hosts can also participate in the protocol. As more agents are available to cover an entire network topology, HPMM returns fault isolation results of finer granularity (as is the case with most management schemes). We do not expect deployment to be a problem as monitoring does not affect performance of the fast path in routers.

Note HPMM's deployement strategy allows it to take advantage of techniques that have been proposed for active networks involving *smart packets* [37], while not requiring deployment of active services. However, the combination of these two techniques is beyond the scope of this paper.

4.1.1 Setup phase

Each agent keeps track of what multicast groups are active in its scope. For routers, the local routing table can be scanned for multicast groups flowing through the router. Hosts can either monitor groups that are relevant only for themselves, or all relevant groups on the local network as a representative for all neighboring hosts. In the former case, local tables are able to provide the necessary information. The latter case is more complicated. One possibility is to query the local router MIB through SNMP. Unfortunately, this only reports the last join/leave seen. It is therefore better to periodically send IGMP membership queries. This means though that the agent must not have joined the monitored group in order to perceive IGMP receiver reports from other hosts. This is achieved by monitoring the group through a packet filter without actually joining the group. One such packet filter is the *Berkeley Packet Filter* [31].

Whenever there is no other active receiver for a group, the agent has to stop monitoring the group — except for the case when an administrator has designated a group to be monitored. For example, if HPMM is setup to monitor test traffic, the agent will purposely join the apropriate multicast group.

Upon settling on the active groups for a node, the HPMM agent must discover an appropriate parent for monitored groups. For a network where HPMM agents appear on each subnet, determining upstream parents is simply configured by announcing to neighbors the TTL of received multicasts. The agent with the highest TTL is most upstream. For cases where agents are not neighbors, several methods of agent organization are possible (without the use of multicast by agents). Several methods that do not require modifications to routers are currently under standardization by the IRTF Reliable Multicast Transport group [14]. A simple scheme is for agents to use mtrace to discover topology information (but not diagnostic

information) to the edge of the domain; next, the HPMM agent sends a unicast message identifying itself to the management station along with the monitored multicast groups and the corresponding path string or distance estimation. The management station acknowledges the registration and hands back the nearest upstream parent for each group. It is also feasible in static routing environments to pre-define parent agents manually. Allowing for router modifications, the best method would be to use *Generic Router Assistance* (GRA) [16] functionality to discover upstream agents. However, we again stress we expect HPMM to be deployed on every subnet simplifying tree construction.

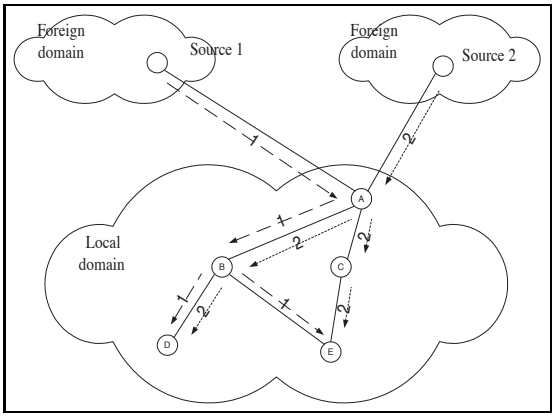


Figure 1: Hierarchical Monitoring

Figure 1 illustrates a completed organization of agents. The HPMM agent at node D has only one parent for both multicast groups 1 and 2, which is node B, while the agent at node E defines a parent agent in B for group 1 and a parent agent in C for group 2. Each agent knows exactly which upstream agent to notify in case of a fault occurrence.

4.1.2 Maintenance phase

After the setup phase, each agent monitors ongoing traffic within each group. To reduce the amount of groups that have to be monitored to provide full network coverage, we introduce the idea of *group representatives* (discussed in detail in Section 4.4). Whenever a fault is perceived within a monitored group, a fault report is sent to the corresponding HPMM parent. Since the parent is monitoring the same group, it can determine if the fault is occurring at its location. If the fault is also present at the parent, the child's report is acknowledged. The parent also signals the fault to the next upstream parent in the same manner. Since each fault is seen by the parent before it reaches the children, children do not have to define a backoff period before they sent a fault report to their parent but can instantly report without feedback implosion.

Eventually, the fault report will reach an agent that does not perceive the same fault. At this location, the process of fault isolation takes place. Through the parent-child relationships, agents can track down the fault cause and location through observable data or where the report came from and whether there are similar incoming reports from other children. In section 4.2 we take a closer look at how this is done for different types of faults. Having isolated the fault location and type, the HPMM agent located just above the fault sends an aggregated fault report to the management station, indicating fault location, type, the groups and children it affects.

All communication, including parent-child messages and fault reports to the management station, is done exclusively through unicast. Each message is explicitly acknowledged by the receiver to ensure correct oper-

ation of the protocol. Reporting through unicast reduces the uncertainty that reporting through multicast would imply. This is also important in terms of possible future deployment of PIM-SSM [10].

HPMM is also designed to reduce the uncertainty of not receiving reports from children. Since it is not clear whether the non-existence of fault reports is due to no fault occurrence or due to a network fault preventing fault reports to reach their destination, *keep-alive messages* have to be sent from children to their parents in certain intervals to acknowledge correct function. This method also enables us to detect network partitioning for multicast transmission.

4.2 Monitoring and Isolation

We detail fault detection procedures with definitions from Section 3 and representative parent node A with two children B_1 and B_2 as shown in Figure 2. Due to space limitations, we only discuss representative faults here. However, note that HPMM is able to detect all faults that can be perceived through SNMP and MIBs, but with the advantage of local fault isolation and scalable reporting.

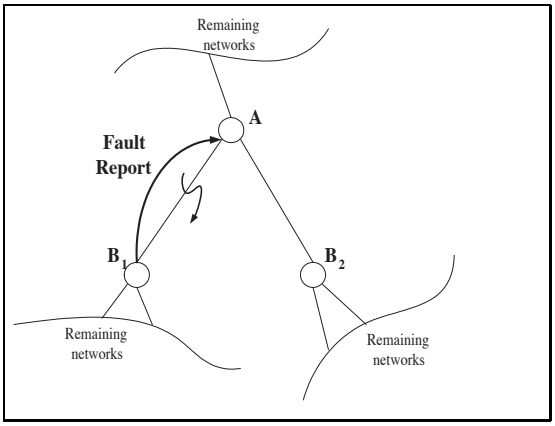


Figure 2: Isolation Scenario

Topological disconnectivity (total/partial) — Assume a connectivity problem between A and B_1: A is not receiving any hello messages from B_1 but has confirmed that B_2 is live. In this case, A can send a fault report to the management station, stating a disconnectivity problem for the area between A and B_1 and all receivers below B_1. Note there is a different results if B_1 has crashed. In that case, HPMM agents below B_1 also perceive the problem of not being able to reach B_1 for reports and hello messages. They also send a fault report to the management station, isolating B_1 as the faulty node. Additionally, all direct children of the crashed node re-enter the setup phase to find the a new parent. In our example, B_1 and B_2 must do this.

Forwarding errors — With a general forwarding problem located at A, both B_1 and B_2 report faults that are not seen at A. Since it is unlikely (but possible) that both links from A to B_1 and B_2 have the same problem, A defines itself as the fault location, with all receivers below it being affected by the problem, and reports this to the management station.

If there is a forwarding problem for only a specific interface of A, say the one to B_1, only B_1 reports the problem. The difference between a malfunctioning interface and a faulty link can be determined by the type of fault in B_1. With an interface problem, whole packets are not transmitted at all. As for a problematic link, packet are most likely to be transmitted, but are dropped at the incoming interface of B_1. Depending on the fault type reported by B_1, A can restrict the problem to either interface or link. In addition, the

agent in A can initiate a local interface check. This is possible for hardware routers. For software agents, this might not be applicable, in which case fault isolation cannot distinguish between link and interface faults.

Packet loss — The perception of packet loss results in a fault report being sent to the corresponding parent. If the parent identifies that the problem exists at its own location, the fault report is sent to its own parent. Eventually, a node will be reached that is not perceiving the problem. This results then in the scenario described in the previous item, except for the fact that packet loss is often caused by queue overflows on incoming interfaces. This can be detected through local statistics, like MIB entries though.

Duplicates — For plain IP multicast, duplicates are detected whenever a specific combination of source address and sequence number has already been received. This means of course that we have to keep track of previously received combinations. This can be problematic for a long history list. If we limit the history window to a small but sufficient amount, we can provide detection for a certain interval range. If so, the method of dealing with this problem is the same as for packet loss. Additionally, if the duplication is the result of the agent receiving packets for the same group on two different interfaces, then it is easy for agents to isolate the fault at their location.

Other — The detection of non-pruning members in the network is not an easy task for HPMM. In a scenario where every entity of the network implements HPMM, the non-pruning member can detect itself by comparing incoming groups with their entries in the routing table.

Without any HPMM agent sitting at or behind the non-pruning member, detection is only possible with complete topology knowledge, meaning that the number of possible multicast users behind the faulty node has to be compared with the multicast groups being monitored at the closet HPMM agent. It is likely that links leading to non-pruning members have a significantly higher number of multicast groups to be monitored than average.

There are multicast faults that do not belong to any of these categories. For example, misconfiguration or incorrect implementation can lead to problems during multicast setup. We have not validated this scenario, but we admit that it might be difficult to use HPMM in the initial phases of multicast setup. We hope to extend our work to this difficulty scenario in the future.

4.3 Passive Monitoring

We define the notion of *passive monitoring* (PM) used in HPMM as agents that supervise only actual network traffic that is flowing in the network without introducing specific test packets for traffic measurement. In contrast, MRM relies on *active monitoring* (AM).

The advantage of AM lies in event pre-testing, but PM can also be used for active testing by having the source application sending test packets previous to the impeding session.

Other than event pre-testing, active monitoring is not well suited to constantly monitor the whole network. First, it introduces additional traffic into the network. Lower latency fault detection requires larger traffic

overhead [7]. Second, tests have to be set up in a static manner; traffic statistics on current flows are not collected. This means that a network administrator has to know multicast flows in the network well enough to set up AM tests effectively.

In contrast, passive monitoring readily adjusts to real traffic. HPMM periodically checks for relevant group memberships. If no receivers are joined to a currently monitored group, monitoring for the group is canceled. PM does not introduce additional traffic other than aggregated fault reports.

PM has a few disadvantages. The most significant is the issue of gathering statistics. With AM, explicitly defining the format of sent data, AM can collect statistics easily (even though just for the test traffic). PM has to deal with the message format used for data as is flows by without the ability to add information. This is especially true for plain IP multicast, where not even a flow-based sequence number is provided in UDP. Therefore, other ways to detect faults have to be used. The easiest scenario is defined by monitoring RTP-encapsulated data, as is done in audio/video sessions. Here, the RTP profile defines session-based sequence numbers, making loss detection simple. It is a reasonable to expect multicast traffic to be transmitted within a RTP packet frame (note that it is not needed to use the whole RTP/RTCP protocol suite, but merely a packet frame encapsulation). Unfortunately, several of today's popular applications, like Real Media or MS Media do not use RTP/RTCP.

Without relying on a specific protocol, HPMM must use local statistics for fault detection. MIB entries can provide information about queue usage, locally dropped packets and packet loss due to failed CRC checks. Through monitoring traffic and calculating average inter-arrival delay and average jitter, packet loss can be estimated if packet arrival delay is a multitude of the expected delay. As already stated in Section 2.1, it has been observed [41] that packet loss in multicast in the internet has almost always a burst-like characteristic. This means that constant, singular packet losses are only minimal, while excessive loss of multiple consecutive packet dominates. Therefore, we assume our estimation to detect most faults.

In the worst case, HPMM can detect all problems associated with statistics that are gathered by legacy SNMP tools, but with the benefit of providing local fault isolation.

Another problem is that HPMM can't distinguish between a long network outage and the end of a transmission, since PM does not have knowledge of when a multicast session ends. Therefore, when a group exists with no traffic present longer than a predefined time for the entire coverage of monitoring agents, a fault report is sent, and then afterwards only every few minutes unless packets begin flowing again. After a preset time, a group without any messages is considered to be ended and therefore not monitored any longer. An improved method might be to have the management station implement a session monitor, e.g., based on sdr [24].

4.4 Group Representatives

Monitoring every multicast group running through a node is too much load to handle, especially for backbone routers. It is more convenient to implement a software package that monitors the data within the slow path of routers. With packets having to be copied to the slow path for HPMM to evaluate them, we want to reduce the number of multicast groups as much as possible.

For this reason, we define *representatives* and group *equivalence classes* (ECs). All groups within an equivalence class provide the same information on network faults for HPMM evaluation. This means that by monitoring more than one group within each EC, HPMM gathers redundant, unnecessary data. It is sufficient to pick one group within each EC to be a representative for this EC. Only the representative has to be monitored. The representative group within an EC can be chosen round-robin over time. We refer to all groups within the same EC *EC-similar*.

We can determine three basic types of EC-similarity, as illustrated by Figure 1.

- *No EC-similarity* — If no EC-similarity can be detected, groups are put into different equivalence classes.
- *Total EC-similarity* — Two groups that have the same source and, wherever they both occur, have the same path back to the source are totally EC-similar. One could also say that they share the same multicast tree within the monitored network. In the example network in Figure 1, multiple groups originating from either source that follow the same path are totally EC-similar.
- *Partial EC-similarity* — This means that two groups are only EC-similar within a certain area of the network. To define rules to detect EC-similarity, we have to clarify our assumption for network faults. First, all traffic flows along a common link share the same fault. Second, multicast forwarding in routers is either correct or faulty for all multicast traffic. We don't expect forwarding faults to be dependent on a combination of incoming and outgoing interfaces. Third, faults in outgoing interfaces affect all traffic flows.

  We acknowledge that the first assumption is not true for all types of faults. Packet loss can vary even between two groups with the same source and set of receivers. Especially in environments with *Random Early Detection* (RED) [23] mechanism, flows have different drop rates along the same link. In this case, we would have to split down the partial EC-similarity into sub-groups of closely related flow characteristics. This is a difficult task, and for this initial version HPMM we restrict packet loss characteristics from EC classification. Each agent defines partial EC-similarity for all groups that come in on the same interface from the same parent agent. This means for our example network, that A has to monitor group 1 and 2, while B and D need to monitor only group 1. C monitors only 2, while E has to supervise both 1 and 2.

  As already noted, representatives can be chosen round-robin over time. This means though that there exists the possibility of the parent's not monitoring the same group than the children. In case of a

fault, the parent perceives a similar problem for its own monitored representative and is therefore able to further isolate the problem. There exists a degree of uncertainty whether the fault matching within one EC is correct. To reduce this, a weighted round-robin scheme is selected, where groups with higher fault rates stay representatives for a longer period than groups with low fault rates. If the difference between the fault rates grows over a given threshold, the EC is split in multiple classes so that fault rates are equivalent within one class.

There are more costly, but failsafe methods to implement the EC scheme. Whenever a parent receives a fault for a group it is not monitoring, it starts to monitor this group additionally, so that further reports can be handled. This certainly increases the report response time for the initial report, but successive reports are processed instantly. The most fail-safe method is to have the parent monitor all groups that its children monitor and that only define partial EC-similarity. Here, we have minimal fault report time. The disadvantage consists in a large number of groups to be monitored in backbone agents.

Further investigation and testing has to be done to verify non-failsafe EC schemes and to select the best method.

4.5 Application of HPMM

HPMM is a protocol that can be included in many existing management system to include multicast management. HPMM provides fault detection, fault isolation and fault reporting for a management station. The visualization of HPMM's fault reports for network administrators can be done within existing management systems, e.g., HP OpenView [27].

HPMM is targeted to monitor multicast flows within one administrative domain, since monitoring agents have to be deployed in network nodes, usually by installing additional software. This is hard to accomplish inter-domain as it requires the cooperation of several organizations; however, technically it is feasible. To provide a level of security for the administrator, a reduced exchange of information between different domains is possible. We plan to follow this thought in later revisions of HPMM.

Another possibility for inter-domain usage is to use HPMM within the *Globally Distributed Troubleshooting* (GDT) protocol [40]. HPMM is well suited as a domain-expertise module for multicast, with GDT providing the framework for automated troubleshooting. Both fault detection and fault isolation experts for multicast can be defined within HPMM.

Besides the usage as a stand-alone protocol, the HPMM mechanism is also valid to be used as an extension for SNMP, so that fault detection and isolation for multicast is handled within the SNMP environment through the usage of HPMM.

Multicast has lately seen an increased use in the area of content distribution networks as an efficient method of distributing popular content to several thousand caches. Often, content servers are arranged in a hierarchy to improve manageability. This environment is almost perfectly suited for a hierarchical

multicast monitoring scheme. And since content servers are often distributed over long distance, localized fault detection reduces maintenance costs and network traffic and reduces the problems associated with monitoring networks across domains (which is different than monitoring problems between domains). Our future work will be to consider this scenario.

Finally, we note that HPMM is designed to work in combination with PIM-SSM or future revisions to multicast routing and the multicast service model since HPMM does not rely on many-to-many multicast delivery. There is a growing expectation that PIM-single source multicast (PIM-SSM) [10] will be widely deployed commercially. In a source-specific environment, only a dedicated source is allowed to send data to the group. Each receiver has to explicitly join both the group and the source by using IGMPv3 source-specific joins [15]. The use of PIM-SSM has two important implications.

1. No native many-to-many transmissions are possible. This prevents the use of monitoring tools like MRM that reply on multicast as a coordinating backoff method to reduce report implosion. Even more drastic is the effect on RTP/RTCP-based tools, since statistical data could no longer be transmitted to all the receivers.

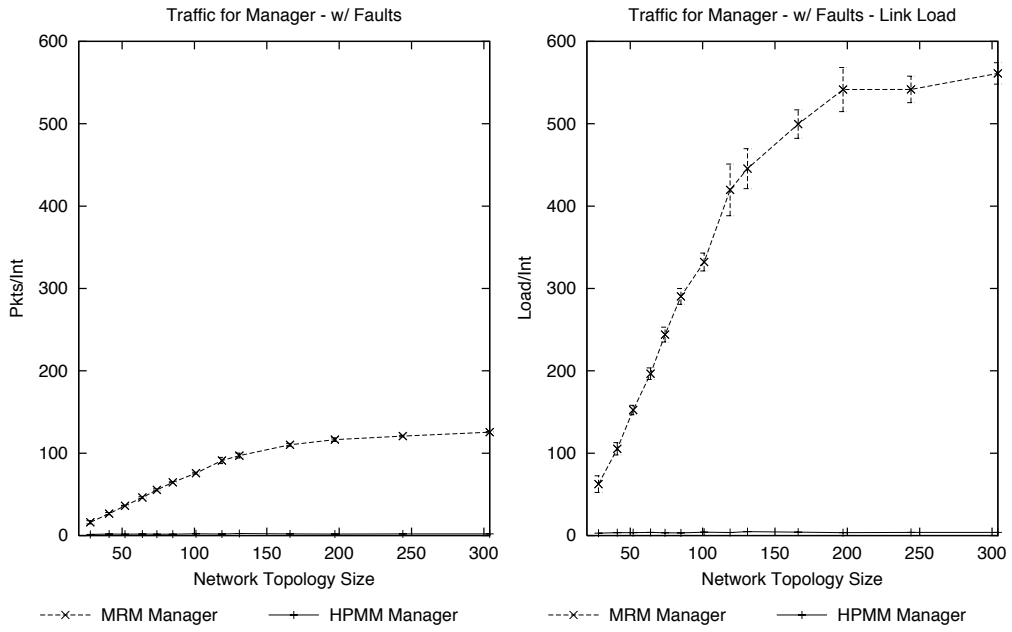
2. It is reasonable in most scenarios to emulate many-to-many transmissions by using multiple one-to-many transmissions. Therefore, the overall number of multicast groups increases, which implies additional burdens on monitoring protocols that already have scaling problems, such as SNMP. Furthermore, special protocols would have to be developed for receivers to learn appropriate addresses assigned to them.

For these reasons, the approach we present in this paper does not rely on many-to-many multicast for fault reporting. In fact, receivers do not use multicast at all.

5 Simulation

To validate our approach, we conducted several simulations. For this purpose, we implemented HPMM and MRM in ns2 [19]. Additionally, basic functionality was also implemented for the approach authored by Reddy et al [35]. Those authors did not name their technique, and for convenience we refer to it as "MTR" approach due to its usage of mtrace messages.

We used the Georgia Tech Internet Topology Modeler (GT-ITM) [42, 43] to produce random graphs for simulation. Our primary purpose was to evaluate performance of monitoring schemes with a growing network topology (and therefore also a growing number of monitoring agents in the network). For each network size, we ran 30 different topologies, varying the average branching factor between 2 and 7 and the average tree depth between 2 and 6. Each topology was run 5 times. All graphs show averages of these values for each network size. A 95% confidence interval was also calculated and is displayed in the graphs. Note that for small values, the interval is too small to be perceived.



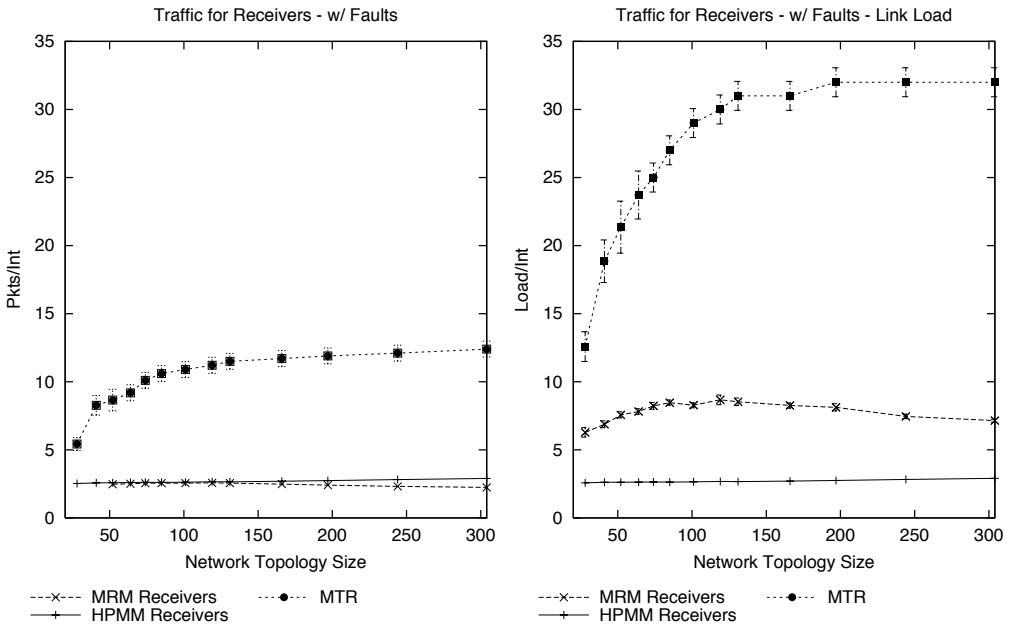
(a) (b)

Figure 3: Average Manager Traffic with Fault Occurrences

MRM supports both fault reporting through multicast and unicast. With multicast reporting, receivers perceiving the same problem backoff as soon as a similar fault report is received. This unfortunately results in reduced fault detection ability, since there is an ambiguity whether a receiver is not reporting due to backoff or due to correct network functions. Furthermore, we assume the use of PIM-SSM, which makes this task difficult. Therefore, we have omitted the case of multicast reporting in our MRM simulation. This means that each receiver is reporting to the MRM management station through explicitly acknowledged unicast.

For MTR, only the subcasting variation was simulated. This method has been identified [35] to provide considerably good performance among other variations of MTR. Although additional router support is necessary, its likeliness of being implemented is higher than for variations dependent on directed multicast services.

Our first set of simulations considers the load placed on the management station and receivers through fault reports. Multicast faults are usually correlated over multiple receivers and therefore the management station is imploded with fault reports. We therefore measured the number of received fault reports at a single management station being responsible for the whole network. At each node, a monitoring agent (either MRM or HPMM) was deployed. With the MTR approach, monitoring is done from the edge nodes. A loss module with uniformly distributed loss rate of 10% was attached to the highest links in the tree,



(a) (b)

Figure 4: Average Receiver Traffic with Fault Occurrences

so that loss correlation areas combined several receivers. The fault report threshold was set to 20%. It might seem we are measuring the worst case behavior of each protocol. However, we ran each simulation for an increasing topology size, effectively simulating a range of fault placements: smaller topologies represent localized faults, large topologies represent faults that affect a larger part of a domain. Each agent reported as soon as a given loss threshold was exceeded. The result is shown in Figure 3(a) for the management station, and in Figure 4(a) for each receiver. As measurement, we used the number of packets per report interval, disregarding the traveling distance of each packet. All packet are of similar size (several KBytes).

The simulations show that MRM introduces considerable traffic at the management station. MRM also results in additional traffic for receivers through test packets. The MTR approach introduces considerable traffic at the receivers, since each mtrace request is subcasted to the receivers below the turn-around router. The interaction of MTR agents and the management station is not specified by Reddy at el. Therefore, we did not measure it.

In HPMM, receivers and management station perceive only minimal traffic.

In these simulations, we expect an SNMP-based multicast management scheme to perform like MRM, since all entities in the network just report on base of thresholds, as they would do with SNMP traps.

In Figures 3(b) and 4(b), we also take into account the traveled distance in router hops of each packet. The longer each packet traveled trough the network, the higher its measured value. We therefore are able to

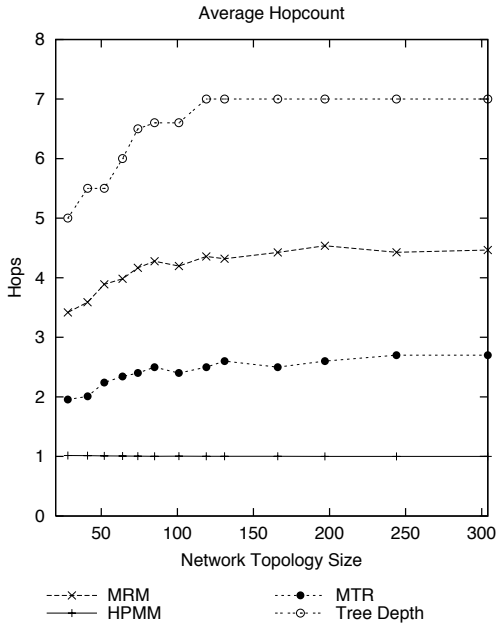


Figure 5: Average Message Hopcount

evaluate the overall link load in the whole network. This also provides insight on the locality of each scheme.

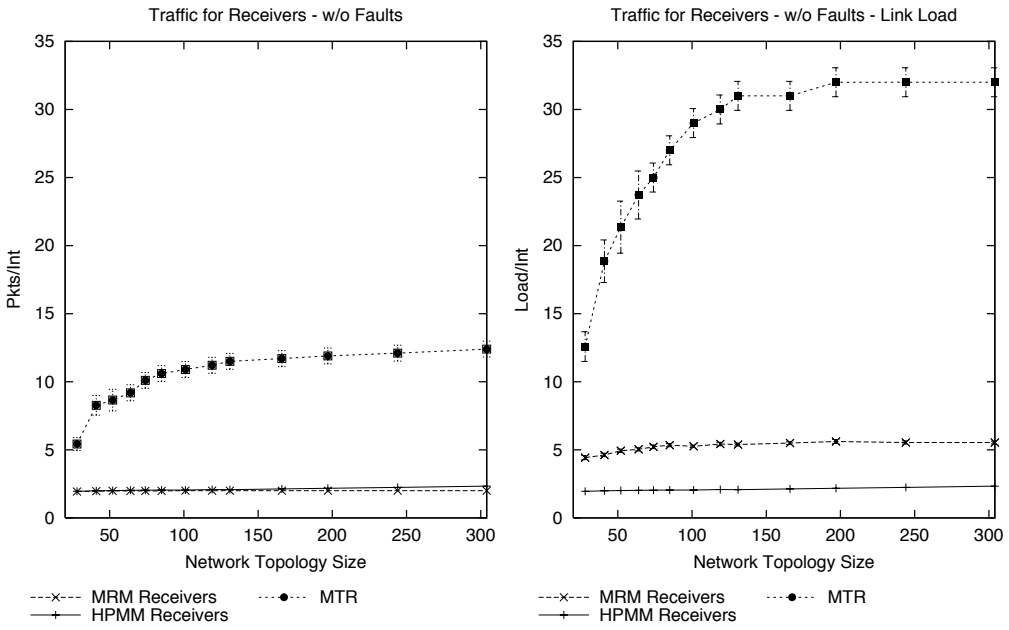
By comparing Figures 3(a) and 3(b), we see that fault reports in MRM travel longer distances than HPMM, since each agents report directly to the management station, which is a single node in the network. Figures 4(a) with 4(b) show that subcasting in MTR also implies that packets have to be sent over multiple hops, although its locality is better than MRM. HPMM performs best in both cases. HPMM scales best with a growing network topology and large deployment of agents.

With HPMM defining a hierarchy of daemons and reducing communication to local areas, it is obvious that most messages in HPMM only travel short distances, in general only to the next hop (if a daemon is deployed at each hop). MRM performs less well, as can be seen in Figure 5.

Of additional interest for a monitoring scheme is the amount of network overhead being introduced through the protocol without any network faults. To determine this value, we took the same topologies, again varied over multiple branching and depth factors, but ran it without any fault occurrences. To compare the three different schemes, we again took a common measurement interval. In Figures 6, we show network traffic for receivers. With no faults, there is no traffic for the management station (except for beacon / keep-alive messages, which are negligible).

MRM's additional traffic contributes to test packets sent to measure faults. MTR implies most traffic overhead of all schemes, since multiple subcasts are conducted. HPMM requires no additional traffic other than keep-alive messages.

There are several remarks to be made for this simulation. It is not trivial to compare these three schemes,



(a) (b)

Figure 6: Average Receiver Traffic without Fault Occurrences

since both MRM and HPMM gather statistics by observing traffic, while the MTR approach gathers snapshots of statistics from routers. With MTR gathering statistics at the same rate as MRM sends test packets, we get a distribution as in Figure 6. In reality, MRM might need more test groups to monitor the network, while MTR might be satisfied with a larger probing interval. So the curves for MRM and MTR might close up to each other. Nevertheless, HPMM shows the best performance, since the only overhead it infers are the keep-alive messages, whose interval is independent of network traffic.

6 Conclusion

Multicast monitoring and management requires additional technology beyond legacy network management systems, which are not able to cope with the complexity of multicast. Additionally, currently available multicast monitoring tools provide the functionality necessary for product-level deployment.

To address this problem, we proposed the *Hierarchical Passive Multicast Monitoring* (HPMM) protocol, a distributed multicast monitoring scheme that uses self-organized monitoring daemons for hierarchical aggregation of fault reports. HPMM results in improved fault detection and local fault isolation over related proposals as it works during network partitions when central reporting stations cannot be contacted, and with PIM-SMM, where receivers are not expected to be multicast capable. We have shown that HPMM

fault reporting scales better with a growing topology and introduces less network traffic than MRM and other schemes. We also discussed fault types and fault correlation in the network and how they can be efficiently detected with HPMM. Furthermore, unlike recently proposed approaches, we have shown HPMM can achieve better performance even without modifications to multicast routing.

Since HPMM requires software daemons to be deployed in the network, it is restricted to intra-domain usage. In future work, We have explored possible ways to use HPMM over multiple domains within inter-domain troubleshooting protocols such as GDT. Similary, we wish to extend HPMM to work between hosts and proxies accross domains for content delivery network monitoring.

HPMM is the first passive monitoring approach, and yet is completely compatible with active monitoring techniques. Additional future work will consider extensions to the basic EC similarity scheme we have introduced in this paper.

References

[1] K. Almeroth. Managing IP multicast traffic: A first look at the issues, tools and challenges. *IP Multicast Initiative Summit*, September 1998.

[2] K. Almeroth. The evolution of multicast: From the MBone to inter-domain multicast to Internet2 deployment. Technical report, UCSB, September 1999.

[3] K. Almeroth. A long-term analysis of growth and usage patterns in the multicast backbone (MBone). Technical report, University of California, Santa Barbara, July 1999.

[4] K. Almeroth and M. Ammar. Collecting and modeling the join/leave behaviour of multicast group members in the MBone. Technical report, Georgia Institute of Technology, 1997.

[5] K. Almeroth and M. Ammar. Multicast group behavior in the Internet's multicast backbone (MBone). *IEEE Communications*, 35:224–229, June 1997.

[6] K. Almeroth and L. Wei. Justification for and use of the multicast routing monitor (MRM) protocol. *IETF Internet Draft*, February 1999. draft-ietf-mboned-mrm-use-*.txt.

[7] K. Almeroth and L. Wei. Multicast reachability monitor (MRM). *IETF Internet Draft*, April 1999. draft-ietf-mboned-mrm-*.txt.

[8] D. Bacher, A. Swan, and L. Rowe. rtpmon : A third-party RTCP monitor. In *Proceedings of the Fourth ACM Multimedia Conference (MULTIMEDIA'96)*, pages 437–438, November 1996.

[9] P. Bhagwat, P. Misra, and S. Tripathi. Effect of topology on performance of reliable multicast communication. In *Proc. IEEE Infocom 94*, pages 602–609, June 1994.

[10] S. Bhattacharyya, C. Diot, L. Giuliano, and R. Rockell. Deployment of PIM-SO at sprint. *IETF Internet Draft*, March 2000. draft-bhattach-diot-PIMSO-*.txt.

[11] J. Bolot and H. Crépin. Analysis and control of audio packet loss over packet-switched networks. Technical report, INRIA, 1993.

[12] J. Bolot, H. Crepin, and A. Vega Garcia. Analysis of audio packet loss in the Internet. In *Proc. 1995 Workshop on Networks and Operating System Support for Audio and Video*, pages 163–174, 1995.

[13] Caida. Caida monitoring. Internet Web Page, 2000. http://www.caida.org.

[14] B. Cain, D. Chiu, M. Kandansky, and B. Levine. Reliable multicast transport building block: Tree auto-configuration. *IETF Internet Draft, to be published*, March 2000. http://www.ietf.org/html.charters/rmt-charter.html.

[15] B. Cain, S. Deering, and A. Thyagarajan. Internet group management protocol version 3 (igmp v.3). *IETF Internet Draft*, August 1995. draft-cain-igmp-*.txt.

[16] B. Cain, T. Speakman, and J. Kurose. Generic Router Assist (gra) building block motivation and architecture. *IETF Internet Draft*, March 2000. draft-ietf-rmt-gra-arch-*.txt.

[17] J. Case, M. Fedor, M. Schoffstall, and J. Davin. Simple network management protocol. *IETF RFC*, 1157, May 1990.

[18] C. Diot, B. Levine, B. Lyles, H. Kassem, and D. Balensiefen. *Deployment Issues for the IP Multicast Service and Architecture*. IEEE Networks Special Issue on Multicast. January/February 2000.

[19] K. Fall and K. Varadhan. ns notes and documentation. Technical report, The VINT Project, UC Berkeley, LBL, USC ISI, Xerox PARC, 1999.

[20] B. Fenner. mrouted 3.9-beta, mrinfo and other tools. Documentation, March 1998.

[21] W. Fenner. Internet group management protocol, version 2. IETF RFC 2236, Xerox Parc, November 1997.

[22] W. Fenner and S. Casner. A "traceroute" facility for ip multicast. *IETF Internet Draft*, March 2000. draft-ietf-idmr-traceroute-ipm-*.ps.

[23] S. Floyd and V. Jacobson. *Random Early Detection gateways for Congestion Avoidance*, pages 397–413. IEEE/ACM Transactions on Networking. August 1993.

[24] M. Handley. Sdr: Session directory tool. Technical report, University College London, March 1995.

[25] M. Handley. An examination of MBone performance. Technical Report ISI/RR-97-450, Information Science Institute (ISI), University of Southern California (USC), January 1997.

[26] Merit Inc. Mview utility. Internet Web Page. http://www.merit.edu/ mbone/mviewdoc/Welcome.html.

[27] Hewlett-Packard Laboratories. mmon multicast management. press release, 2000. http://www.hpl.hp.com/mmon.

[28] B. N. Levine, S. Paul, and J. J. Garcia-Luna-Aceves. Organizing multicast receivers deterministically according to packet-loss correlation. In *Proceedings of the 6th ACM International Conference on Multimedia (Multimedia-98)*, pages 201–210, September 12–16 1998.

[29] F. LoPresti, N. G. Duffield, J. Horowitz, and D. Towsley. Multicast-based inference of network-internal delay distributions TITLE2:. Technical Report UM-CS-1999-055, University of Massachusetts, Amherst, Computer Science, November, 1999.

[30] D. Makofske and K. Almeroth. *MHealth: A real-time graphical multicast monitoring tool for the MBone*. Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV). June 1999.

[31] Steven McCanne and Van Jacobson. The BSD packet filter: A new architecture for user-level packet capture. In *Proceedings of the Winter 1993 USENIX Conference: January 25–29, 1993, San Diego, California, USA*, pages 259–269, Winter 1993.

[32] K. McCloghrie, D. Farinacci, and D. Thaler. Internet group management protocol MIB. *IETF Internet Draft*, January 2000. draft-ietf-idmr-igmp-mib-*.txt.

[33] K. McCloghrie, D. Farinacci, and D. Thaler. IPv4 multicast routing MIB. *IETF Internet Draft*, January 2000. draft-ietf-idmr-multicast-routmib-*.txt.

[34] S. Paul et al. Reliable multicast transport protocol. *IEEE Journal on Selected Areas in Communications*, 15(3):407–421, April 1997.

[35] A. Reddy, D. Estrin, and R. Govindan. Fault isolation in multicast trees. In *Proc. of ACM Sigcomm 2000*, January 2000.

[36] Schulzrinne, Casner, Frederick, and Jacobson. Rtp: A transport protcol for real-time applications. *IETF Draft*, November 1994. draft-ietf-avt-rtp-*.txt.

[37] Beverly Schwartz, Alden W. Jackson, W. Timothy Strayer, Wenyi Zhou, Dennis Rockwell, and Craig Partridge. *Smart Packets for Active Networks*, pages 397–413. ACM Transactions on Computer Systems. February 2000.

[38] D. Thaler. Globally-distributed troubleshooting (gdt): Protocol specification. *IETF Internet Draft*, September 1997. draft-thaler-gdt-spec-*.ps.

[39] D. Thaler and B. Aboda. Multicast debugging handbook. *IETF Internet Draft*, Oktober 1998.

[40] D. Thaler and C. V. Ravishankar. An architecture for inter-domain troubleshooting. Technical Report CSE-TR-344-97, University of Michigan Department of Electrical Engineering and Computer Science, August 1, 1997.

[41] M. Yajnik, J. Kurose, and D. Towsley. Packet loss correlation in the MBone multicast network. Technical Report 96-32, University of Massachusetts, Amherst, 1996.

[42] E. Zegura, K. Calvert, and S. Bhatttacharjee. How to model an Internetwork. Technical report, College of Computing, Georgia Institute of Technology, Atlanta, GA, 1996.

[43] E. Zegura, K. Calvert, and M. Doar. Modeling Internet topology. Technical report, College of Computing, Georgia Institute of Technology, Atlanta, GA, 1999.