A Hierarchical Multicast Monitoring Scheme

Joerg Walz
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
jgwalz@gmx.de

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

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; and no modifications to existing routing or diagnosis protocols are required. The tradeoff of using HPMM is it leverages a large number of software daemons deployed through out a local domain. We compare the signalling overhead 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 the problems that have deterred multicast deployment [18] is a lack of monitoring and management tools and systems. The successful deployment of network services requires that administrators are able to supervise their correct functioning and to detect and isolate faults. For unicast services, network management systems are able to track and monitor traffic flows in various ways. Supervising multicast traffic is a more difficult problem as each multicast tree involves multiple hosts with correlated, simultaneous faults; furthermore, it is unknown how often faults are correlated over multiple multicast routes. To monitor all ongoing multicast transmissions efficiently, whether within one administrative domain or over multiple domains, an administrator has to use a multitude of different tools. In this paper, we offer a new intradomain multicast management tool called the *Hierarchical Passive Multicast Monitor (HPMM)* that employs a more scalable approach than previous solutions to multicast management.

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 group — 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 offers a framework for monitoring intradomain multicast flows. MRM defines protocols for remotely managing testing agents and the collection of fault reports.

MRM is expected to be used by *actively* injecting supervised 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 which *passively* monitors ongoing multicast traffic in the network. Because HPMM is hierarchical and passive, fault messages aggregation and intra-domain fault detection and isolation is provided more efficiently. HPMM satisfies a number of design goals: scalable reporting; fault isolation; no dependencies on multicast routing for reporting; and no modifications to existing routing or diagnosis protocols (such as IGMP mtrace). The tradeoff of using HPMM is it leverages a large number of software daemons deployed one each in multicast-enabled subnets in a local domain. Such daemons are intended to be co-located with SNMP installations and are not meant as router modifications, but rather as application-layer software. HPMM may be viewed as an

NGC '00 11/00 Palo Alto, CA, USA
© 2000 ACM ISBN 1-58113-312-X/00/0011...$5.00

alternative approach for use within the MRM framework.

This paper is arranged as follows. In Section 2, we provide justification for the necessity of multicast monitoring and review previous work in multicast management. Section 3 discusses details of the operation of HPMM, while Section 4 gives an overview of benefits and problems of HPMM. Considerations of applicability are included in Section 5. In Section 6, we present simulations that show the general approach taken by HPMM is able to monitor all multicast traffic within an administrative domain without excessive additional traffic. Our comparisons show the hierarchical and passive approach employed by HPMM has less network overhead than active approaches that can be used with MRM as well as recent monitoring proposals [35] that work outside the MRM framework and require modifications to multicast routing protocols. We offer concluding remarks in Section 7.

2. BACKGROUND

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 techniques is a necessary component for successful large-scale multicast deployment. Whenever faults occur, they must be accompanied by fast and accurate fault detection and isolation. Existing tools fail in this capacity [6].

Monitoring of multicast routing faults can be difficult because faults are correlated among multicast receivers and may result in an explosion of fault reports. Previous work [39, 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 up to 20% of receivers [39], even across domains. Legacy unicast management tools and protocols, like SNMP, do not help management stations in differentiating between correlated and uncorrelated faults. Additionally, protocols like SNMP can flood the management station with reports if a fault affects multiple hosts. This problem is not limited to SNMP: every management scheme based on a central station managing unaggregated reports faces this unscalability (see Section 6 for related simulation results). *The goal of multicast monitoring tools and protocols is therefore to reduce the number of fault reports without reducing fault response time, and to minimize the amount of work required to detect and precisely isolate faults.* These tasks may increase in difficulty with the size of the monitored network and the number of concurrent multicast sessions.

2.2 Related Work

Monitoring approaches for multicast traffic have generally 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. One exception is 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 has provided an excellent overview of multicast evolution [2].) During this phase, debug-level tools were developed to verify specific problems in early multicast usage. Although the functionality of these tools were later expanded, they are still not intended to be used by a non-expert user.

The most popular 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. Similarly, *mrinfo* [20] 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 and Aboda [38].

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 (by Bacher, Swan, and Rowe [8]), and MHealth (developed by Makofske and Almeroth [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 these 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. A popular management system using this technique is HP Openview, with multicast extensions having only recently been added to its functionality with a prototype application called *mmon* [27]. For inter-domain usage, proprietary software agents are used that run at pre-defined nodes and report to a central management station. The CAIDA (Cooperation Association for Internet Data Analysis) group [13] has developed such 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. This section describes MRM in some detail, however, readers are encouraged to consult the full protocol specification [7] for a definitive description of MRM.

MRM can be deployed at hosts or routers. It 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 a *test receiver* (TR). The functionality of MRM includes the following:

1. The 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. The initiation of multicast transmissions by test senders to a specified multicast test address.

3. The 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 6 that MRM does still not scale well enough for large-scale deployment. Furthermore, MRM can be configured to use multicast itself to report faults. This implies both the problem of lost reports during multicast outages and reduced fault detection ability. Nevertheless, we believe MRM is a seminal step towards a solution to multicast monitoring as it does not rely on unicast management protocols and attempts to address the problem of multicast monitoring within one protocol.

The last approach we review here has been recently introduced by Reddy, Estrin, and Govindan [35] [1]. Their technique isolates faults using IGMP MTRACE messages initiated by receivers and then gathers statistics. So that every receiver trace does not implode at the multicast source, MTR introduces 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 less practical and less accurate as compared with other approaches. 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. Finally, we show in Section 6 that even with router modifications, this scheme does not operate as efficiently as the approach we introduce in this paper nor does the approach operate as efficiently as MRM, although MRM and HPMM do not require such modifications. In addition, MTR lacks specification of how to handle the implosion of reports from receivers when a fault is detected, it is not proven to detect multiple simultaneous faults [35], and lacks considerations on how multiple multicast groups can be handled efficiently.

3. HPMM PROTOCOL DESCRIPTION

In this section, we define our proposal, called *Hierarchical Passive Multicast Monitoring* (HPMM), which monitors and detects faults in ongoing multicast traffic. We present the protocol specification and fault isolation mechanisms.

We expect HPMM to be deployed in *intradomain scenarios only*, monitoring and isolating only faults that occur locally. Our future work will include how an HPMM-like approach may be used by larger networks, such as carriers, or at POPs carrying high-rate traffic.

HPMM relies on software monitors hierarchically arranged according to network topology and existing multicast routes. The software agents are not modifications at routers, but application-level daemons that may operate in a router's *slow-path* co-located with SNMP operations. 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 has 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 and the other agents involved. 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.

3.1 Protocol Basics

HPMM defines two generic entities: the *HPMM Management Station* (HMS) and *HPMM Monitoring Agents* (*agents*, for short). The HMS is a central facility, usually located in the *Network Operation Center* (NOC). It acts as the interface between the network administrator and HPMM, providing information on gathered statistics and received fault reports. It also enables remote configuration of monitoring agents. The primary tasks of agents consist of monitoring traffic, maintaining fault statistics, building of hierarchical organization and reception and sending of fault reports.

All communication between agents and the HMS, and among agents is exclusively unicast. Reporting through unicast reduces the uncertainty that reporting through multicast would imply. Each message is explicitly acknowledged by the receiver to ensure correct operation of the protocol; lost messages are recovered through retransmissions.

3.2 Hierarchical Setup

The construction of the logical hierarchy between agents, i.e., the definition of child-parent relationships, is the most important part of HPMM. These relationships are used to transmit *fault reports*, *keep-alive* messages, and consequently provide localized fault-isolation mechanisms. A *parent* is the next upstream HPMM agent in a multicast group.

Each agent has to identify the parent agent for each *active* group. Active multicast groups are those where agents are receiving packets. The identification of active groups within the monitoring scope is described in Section 4.2. For now,

[1] The authors did not name their technique, and for convenience we refer to it as "MTR" approach due to its usage of mtrace messages.

we assume that the agent has correct knowledge of groups to be monitored.

It is feasible in static routing environments to pre-define parent agents manually, but this approach is likely to be impractical. To establish child-parent relationships dynamically in HPMM we define the following protocol based on recent standards proposals by the IRTF Reliable Multicast Transport (RMT) group [14]. How an agent, A detects its parent, P, for a group, m, is summarily described below:

1. A discovers its *location information* relative to the source of m. (The source of a group is explicitly available as part of PIM-SSM address.)
2. A sends a subscription request to the management station, along with location information and the address of group m. The management station returns next valid upstream parent P.
3. A adds m and its parent P to its group table. An initial *keep-alive* message is sent to P to subscribe as a new child agent for group m.

More specifically, each step proceeds as follows.

Step 1 — In order to identify its own location relative to the source of m_1, the agent notes the hopcount of received packets for this group by extracting the *Time-To-Live* (TTL) entry from a packet header. Since we expect HPMM agents to be deployed at each subnet co-located with SNMP installations (a cost tradeoff designed into HPMM's assumptions), the agent first tries to resolve the upstream parent within the local subnet by broadcasting these values (with a TTL of 1) on a well-known broadcast address. Neighboring agents on separate subnets bearing a larger TTL value respond to this announcement. This enables A to determine the parent agent without assistance from the management station.

If no parent can be identified by this method — for example is HPMM is incrementally deployed — A gathers additional source path information. For this purpose, we rely on any of the topology discovery mechanisms defined by the IRTF RMT working group [14]. Those include *Generic Router Assistance* protocol (GRA) [16], which may soon be commonly deployed. Where GRA is not available, an IGMPv2 mtrace request [22] can used. Note that we facilitate mtrace only for path discovery, not for acquisition of statistics.

Step 2 — Depending on the outcome of Step 1, agent A either notifies the the management station of a discovered parent P, or requests allocation of a valid parent. (Either action conforms to the IRTF standard for tree building [14].) In the first case, the HMS simply acknowledges the information and updates its database. Thereby, it registers both A's monitoring group m and the child-parent relationship between A and P. In the latter case, the HMS consults its database of already subscribed agents respective to the (source,m) pair information provided in the request message. The resulting parent is noted in the database and transmitted to A.

Step 3 — Agent A establishes a child relationship with P by sending a notification to P and awaits acknowledgments from P. This final step concludes the setup and initiates the actual monitoring in agent A.

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.

Since multicast groups are subject to changes, additions and deletions of active groups are handled accordingly. Child-parent relationships for new groups are found by reentering the hierarchical setup, while the cancellation of a previously existing group results in the removal of the corresponding relationships. The removal is straight forward and not detailed here.

HPMM defines intervals during which additions and removals of relationships cannot occur. This makes the protocol robust in the presence of frequent group joins or leaves by users and multicast route flapping.

Last, the hierarchical structure is maintained through *keep-alive messages* that are sent from children to their parents and from parents to children in regular periods. Thereby, HPMM reduces the uncertainty of not receiving reports from children or acknowledgments from parent; without such keep-alive message, the lack of faults and fault reports is indistinguishable from fault reports that are prevented from reaching their destination due to faults. This method also enables HPMM to detect network partitions.

3.3 Monitoring and Isolation

3.3.1 General methodology

Each agent monitors ongoing traffic for active groups to ensure multicast services are live and without fault (see Section 4.2 on reducing the number of monitored groups). 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. Children do not delay in reporting faults to parents. Similarly, parent's do not wait for children's reports before reporting to their own parents. This is because a parent's fault is always also present at descendents.

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. 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, and the groups and children it affects.

3.3.2 Fault isolation scenarios

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. Due to space limitations, we discuss only representative faults here. The fault definitions primarily follow definitions provided by Almeroth [7] and Thaler and Aboda [38]. In the following examples, we define a parent node A with two children B_1 and B_2.

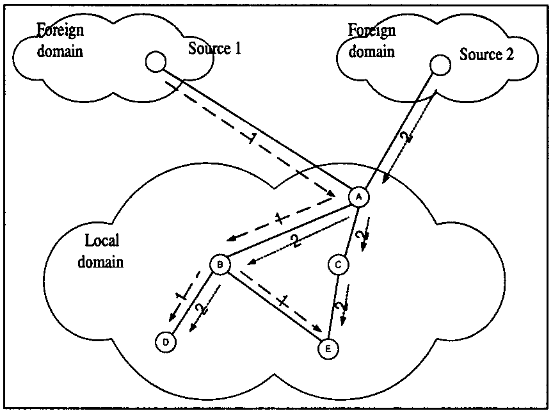


Figure 1: Hierarchical Monitoring

Topological disconnectivity (total/partial) — Assume a connectivity problem exists between A and B_1: A is not receiving any keep-alive 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 presumably all receivers below B_1). Note there is a different result if B_1 has crashed. In that case, HPMM agents that are children of B_1 also perceive the problem of not being able to reach B_1 for reports and keep-alive 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 a new parent.

Forwarding errors — With a general forwarding problem located at A, both B_1 and B_2 report faults that are not seen at A. It is 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 at 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 — Packets loss as caused by queue overflows on incoming interfaces can be monitored by HPMM. Agents maintaining a child-parent relationship are able to pin fault location by comparing average packet loss rates, as can be obtained through local statistics, like MIB entries for example. With a problem between A and B_1, average loss rates show significant discrepancies.

Duplicates — Duplicate packets are usually generated by inconsistencies in router forwarding tables. For plain IP multicast, duplicates are detected whenever a specific combination of source address and sequence number has already been received. For this reason, HPMM agents keep a history list of already seen packets. Depending on the actual configuration, this enables the agent to detect duplicates for a given interval range. For agents located at routers, detection of duplicate routes is also noted as receiving packets for the same group on two different interfaces. The fault isolation is then initiated by contacting the assigned parent agent.

Non-pruning — 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 difficult scenario in the future.

3.3.3 *Identifying a session end*

HPMM cannot distinguish between a long network outage and the end of a transmission just on base of observing multicast traffic, since it has no knowledge of when a multicast session ends. Therefore, when a previously monitored group exists in routing tables (or with current IGMP subscriptions) but without recent traffic, a fault report is sent, and then afterwards only every few minutes unless packets begin flowing again. (An improved method might be to have the management station implement a session monitor, e.g., based on sdr [24].)

4. PASSIVE MONITORING

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 of multicast packet loss correlation [28]; see also performance results in Section 5. The use of passive monitoring in HPMM is perhaps more controversial. In this section, we clarify and propose solutions to difficulties connected with passive monitoring and HPMM in comparison with previous approaches.

4.1 Comparison

While HPMM easily could be based on test traffic, its fault detection mechanisms are designed for passive monitoring of actual traffic. This approach provides several advantages as compared to active injection of test traffic. With passive monitoring, the manual setup of test traffic is avoided. It is not a simple task to define efficient mapping of test traffic for all network paths in current use. Covering the network topology completely may result in a large amount of test traffic overhead. Test traffic introduces additional traffic into the network, even if no fault is detected, and lowering fault detection latency requires larger traffic overhead [7]. Furthermore, static test scenarios may not detect problems with current paths. Passive monitoring readily adjusts to real traffic and therefore simplifies deployment and maintenance. There is no possible discrepancy between faults and gathered statistics of test traffic and actual network conditions. Passive approaches introduce no traffic if no fault is seen, except for messages necessary for hierarchical setup and periodical keep-alive messages, which can be negligible in implementation. Lastly, we note that passive monitoring operates constantly, whereas monitoring of test traffic occurs only when such tests are executed.

On the other hand, passive monitoring presents several difficulties.

First, passive monitoring must be able to work with whatever message format is used with data as is flows by without the ability to add information. Even plain IP multicast may cause problems as no flow-based sequence number is provided in UDP, which may cause difficulties in detecting some faults, such as packet loss. Passive monitoring of RTP-encapsulated data presents the easiest scenario, and it is commonly used in multimedia sessions. RTP defines session-based sequence numbers, making loss detection simple. It is reasonable to expect multicast traffic to be transmitted within a RTP packet frame (note using the whole RTP/RTCP protocol suite is unnecessary but merely a packet frame encapsulation). Unfortunately, several of today's popular applications, like Real Media or MS Media do not use RTP/RTCP.

Second, high-rate traffic can be very difficult to monitor. When the number of active multicast groups is large, an HPMM agent may become overwhelmed with the task to evaluate all packet headers. To address this problem, we introduce below a scheme to reduce the number of monitored groups.

For these two reasons, packet loss (and skips in sequence number) can be the most difficult fault to detect. One positive note is that packet loss in multicast sessions on the Internet have been observed to be commonly bursty [39]. We expect singular packet losses to be rare and excessive loss of consecutive packets to dominate, lessening the possibility of not detecting loss. Monitoring MIB entries regarding queue usage and overflow, locally dropped packets, and packet loss due to failed CRC checks may be easier than detecting skips in sequence number.

Finally we note passive monitoring can rely only on statistics and tools that are available local to an installation for fault detection. In the worst case, HPMM can detect all problems associated with statistics that are gathered by colocated SNMP tools, but with the benefit of providing localized fault isolation.

4.2 Group Selection and Reduction

HPMM agents periodically check for relevant group memberships to determine active groups within the monitoring scope. It is important to dynamically adjust the hierarchy according to the altered multicast distribution. If no receivers are joined to a currently monitored group, monitoring for the group is canceled, while a newly subscribed group must be included in the monitoring process. HPMM delays canceling of monitored groups to avoid overhead due to frequent membership changes. Additionally, we allow network administrator to manually define *include* and *exclude* lists for groups that are required to be monitored or unmonitored.

4.2.1 Determining active groups

For both hardware and software routers, the detection of locally attached group participants is fairly simple. The existence of active groups can be easily determined by examining local routing tables. Routers manage group participation through the *Internet Group Management Protocol* (IGMP) [21], which defines Host Membership Queries, that can be used to locate active receivers for a group on the local network.

For HPMM agents running at host locations, local routing tables are not able to provide information about other receivers in the same subnet. Therefore, we recommend the agent having SNMP access to the local router so that queries to the router's IGMP MIB entries are possible. Periodically, the IGMP MIB entry *IGMP cache table* is requested, providing information on active groups. Where SNMP is not possible, or the IGMP MIB is not implemented in the router, the agent has to listen to IGMP messages on its own, i.e., catch join/leave messages, and periodically send IGMP Host Membership Queries.

Agents cannot be allowed to join monitored multicast groups with IGMP since this would interfere with active group detection. HPMM is designed for maximal deployability, restricting host modifications to existing protocols, so we can't modify the host's IGMP behavior. Therefore, a HPMM agent located at a host has to monitor groups through a packet filter on a network interface in promiscuous mode without actually joining the group; the *Berkeley Packet Filter* [31] is suitable for this purpose.

4.2.2 Equivalence Classes

Monitoring every multicast group running through a router is too heavy a workload to manage without disrupting traffic even in the intradomain scenarios HPMM is intended for. Instead, HPMM should be implemented as software in the slow path of a router monitoring only a subset of multicast groups.

To reduce the number of groups that must be monitored while providing complete coverage, we define *multicast-fault representatives* (mfrs) and group *multicast-fault equivalence classes* (mfecs). All groups within an fault equivalence class are defined to be subject to the same network faults. By monitoring more than one group within each mfec, HPMM gathers redundant, unnecessary data. It is sufficient to pick one group within each mfec to be a representative. The representative group within an mfec can be chosen round-robin over time. We refer to all groups within the same mfec as *mf-similar*.

Such an approach is implicitly taken by other monitoring tools. Test traffic as used with MRM forms the basis of fault detection for all actual traffic that shares the same path. Using the terms introduced in this section, all groups on this path are implicitly combined in one mfec with the test group as the mfr. HPMM reduces monitored groups on a more localized scope: each pair of agent peers determines their own equivalence classes and representatives. The closer two agents are, the more likely a fault between these two affect all multicast flows running through these agents.

We define three basic types of mf-similarity, as illustrated by Figure 1.

- *No mf-similarity* — If no mf-similarity can be detected, groups are put into different equivalence classes.
- *Total mf-similarity* — Two groups that have the same source and, wherever they both occur, have the same path back to the source are totally mf-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 mf-similar.
- *Partial mf-similarity* — This means that two groups are only mf-similar within a certain area of the network. To define rules to detect mf-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.

Our first assumption regarding partial mf-similarity may not be 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 mf-similarity into sub-groups of closely related flow characteristics. This is a difficult task, and for this initial version HPMM we exclude packet loss characteristics from defining mfec classification. Each agent defines partial mf-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.

Representatives can be chosen round-robin over time by agents; this means there exists the possibility of parents not monitoring the same group as their children. If a fault occurs, the parent attempts to determine if a similar problem for its own monitored representative and able to further isolate the problem. A weighted round-robin scheme can be used 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 mfec is split in multiple classes so that fault rates are equivalent within one class. However, in this initial work we have not verified the robustness of such an approach: there exists a degree of uncertainty whether the fault matching within one mfec is correct.

A more costly but more robust method can be used to implement the mfec 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 mf-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 is required to verify the correct operation of non-failsafe mfec schemes and report on their performance.

For now we restrict HPMM to partition multicast-fault classes depending on topology, not on flow characteristics. This means that as long as two multicast groups share the same path from a parent agent A to a child B, the agent at B defines one class for both groups, limiting monitoring to only one of these, neglecting different flow characteristics. The monitored group is then labeled as mfr and fault statistics for the mfr are equivalent for all multicast groups within the same mfec.

5. DEPLOYMENT ISSUES

HPMM is targeted to monitor multicast flows within one administrative domain, since monitoring agents have to be deployed in network nodes by installing additional software. Intra-domain deployment through co-location with SNMP installations is the most reasonable scenario. In order to isolate faults in the network with fine granularity, HPMM must be deployed throughout a network topology (as is the case with most management schemes).

HPMM can be combined with many existing management system to include multicast management, most notably with MRM. 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].

We note that HPMM is designed to work in combination with *Protocol Independent Multicast - Source-Specific Mode* (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-SSM [10] will be widely deployed commercially. In a source-specific environment, only a dedicated source is allowed to send data to the group (commonly referred to as one-to-many transmission). 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 statis-

tical data could no longer be transmitted to all the receivers.

2. The use of many-to-many transmissions must be replaced with multiple one-to-many PIM-SSM groups, increasing the number of multicast groups in use, which implies additional burdens on monitoring protocols that already have scaling problems, such as SNMP. For monitoring that requires agents to multicast results, protocols would have to be developed for agents to learn the addresses assigned to them to announce reports.

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

For reporting problems across domains, the *Globally Distributed Troubleshooting* (GDT) protocol uses a tree-like structure of *expert location servers* [37] similar to HPMM. GDT is used for automated troubleshooting in inter-domain areas and relies on *domain-specific expertise modules* for fault detection and isolation. HPMM is well suited as a domain-expertise module for multicast in an intra-domain environment, providing both fault detection and isolation experts. With GDT providing the framework for automated troubleshooting and inter-domain knowledge exchange, a combination of HPMM and GDT could prove to be an achievable solution.

Notably, the combined use of GDT and HPMM is not a solution for managing faults in carrier backbones. High traffic rates and overwhelming numbers of concurrent multicast routes are the toughest environments for passive monitoring. Practical issues may also limit the deployment of management tools, such as space at points-of-presence (POPs) maybe limited for placing hardware supporting monitoring agents.

5.1 Further applications

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 from monitoring problems between domains). Our future work will be to consider this scenario.

6. SIMULATION

To explore the performance advantages of hierarchical reporting and passive monitoring, we conducted several simulations. We simulated a simplified HPMM-like protocol in the ns2 simulator [19], ignoring tree construction, mfec-similarity algorithms, and other details. We also simulated a simplified version of an active-testing, non-aggregated fault reporting protocol, as has been previously proposed for the MRM framework. Additionally, we simulated a simplified protocol based on the approach taken by the MTR protocol [35].

Our purpose was to evaluate overhead of monitoring schemes with a growing network topology (and therefore also a growing number of monitoring agents in the network). For each of the three approaches we compared, we analyzed the traffic overhead and the link load at agents and central collection stations due to monitoring and reporting for scenarios when faults were and were not presents.

6.1 Assumptions

We executed each simulation described in this section over increasing network topology sizes. For each network size depicted in the graphs, we executed our simulations over 30 different topologies, varying the average branching factor between two and seven and the average tree depth between two and six. We used the Georgia Tech Internet Topology Modeler (GT-ITM) [40, 41] to produce random graphs. The simulation was executed over each generated topology five times with random variations in fault occurrences. All data points displayed in the various graphs of this section show values averaged over all topologies generated of a particular network size. A 95% confidence interval was also calculated and is displayed in the graphs. Note that sometimes the interval is too small to be perceivable.

MRM can support fault reporting through multicast or 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 simulations of MRM-based active-testing, non-aggregated reporting. This means that each receiver is reporting to the management station through explicitly acknowledged unicast.

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

It is not trivial to compare these three schemes, since both MRM and HPMM gather statistics by observing traffic, while the MTR approach gathers snapshots of statistics from routers. To be able to compare these three protocols despite their diversity, we measured the overhead of the protocols for a *monitoring period.* This normalized measurements of each approach: rather than count the number of packets generated over the timed duration of the simulation, we instead counted *the number of packets per report interval of the protocol*, which we believe created a fair comparison. For example, with MRM, this is defined as period between two subsequent test packets and therefore between subsequent reports. In MTR, the periodicity of mtraces measures the monitoring period. Finally, HPMM has to periodically inform parent agents of fault status. We effectively set these monitoring periods to the same interval, equalizing the time required for individual agents in each protocol to detect faults. The results in this section must be examined with this normalization in mind. In reality, for example, MRM numbers may be affected by the number of test groups needed to monitor the network, while MTR might be satisfied with a larger probing interval.

Finally, we note these results are of the overhead due to network signaling between agents, not of computational

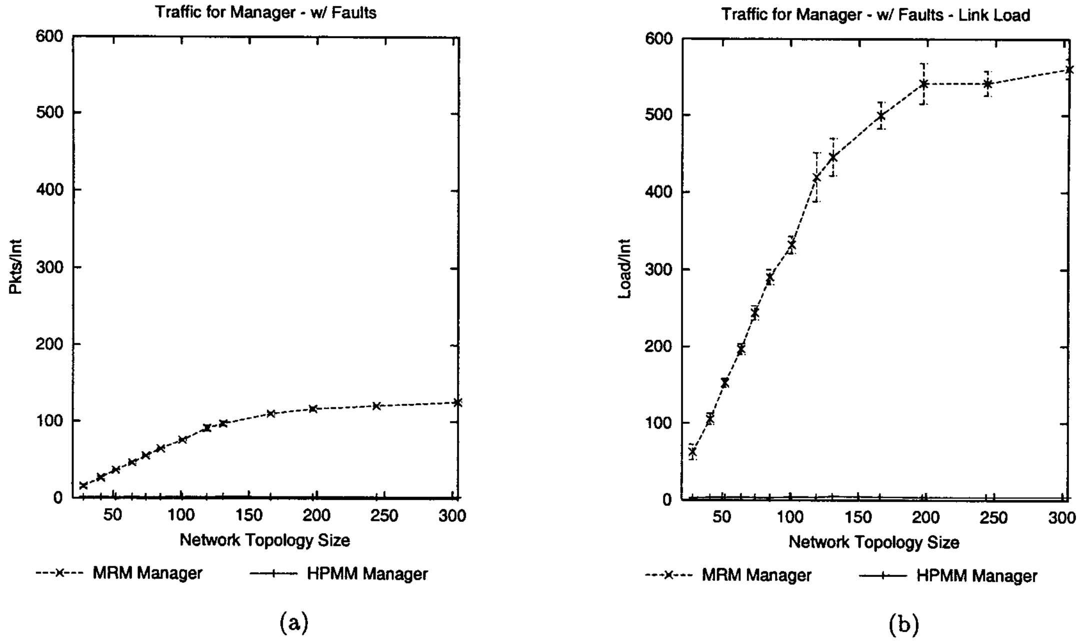


Figure 2: Average Manager Traffic with Fault Occurrences

overheads at each agent. We believe computational overhead would be better determined through the examination of actual implementations and specific hardware.

6.2 Performance Results

Our first set of simulations considers the load placed on the centralized management station and receivers due to the reception of fault reports. All fault report packets sent over the network are the same size (several KBytes) in our ns2 simulation; however, note the graphs show the number of packets, not the bandwidth of the reports. (We assumed fault reports fit in one packet for all protocols.)

A monitoring agent was deployed at each node in the topology that followed the rules of either passive-hierarchical management or active-non-aggregated management. With the MTR-like approach, monitoring is performed solely from the edge nodes.A loss module with uniformly distributed loss rate of 10% was attached to the highest links in the tree, so that loss correlation areas combined several receivers. The fault report threshold was set to 20%. Each agent reported without delay whenever a given loss threshold was exceeded. We measured the number of received fault reports at a single management station responsible for the whole network during this scenario.

By placing the faults at the top of the multicast tree covering the entire topology, 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. In other words, the simulations represent the size of the network topology underneath a fault.

The simulation results shown in Figure 2(a) graph the amount of traffic received by a management station, and in Figure 3(a) the amount of traffic received by each receiver/agent.

The simulations show that an MRM-based active-testing, non-aggregated approach (labeled simply "MRM") introduces considerable traffic at the management station, and also results in additional traffic for receivers through test packets. In these simulations, we expect an SNMP-based multicast management scheme to perform similarly, since all entities in the network report to the central station whenever a SNMP trap threshold is crossed.

The interaction of MTR agents and the management station has not been specified [35]; therefore, we did not measure it. The MTR-based approach introduces considerable traffic at the receivers (larger than the suggested bandwidth of test traffic suggested by the MRM standard), since each mtrace request is subcasted to the receivers below the turnaround router.

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

Figures 2(b) and 3(b) show evaluations of the overall link load of the protocols for the same simulations by taking into account the distance traveled by each packet in router-hops. The longer each packet traveled through the network, the higher its measured load. This metric also provides insight into the locality of each scheme.

By comparing Figures 2(a) and 2(b), we see that MRM-based non-aggregated fault reporting creates a larger link load than than the hierarchical reporting approach taken by HPMM. Figures 3(a) and 3(b) show that although subcasting in MTR causes packets to be sent over multiple hops, its message locality is better than active testing. HPMM

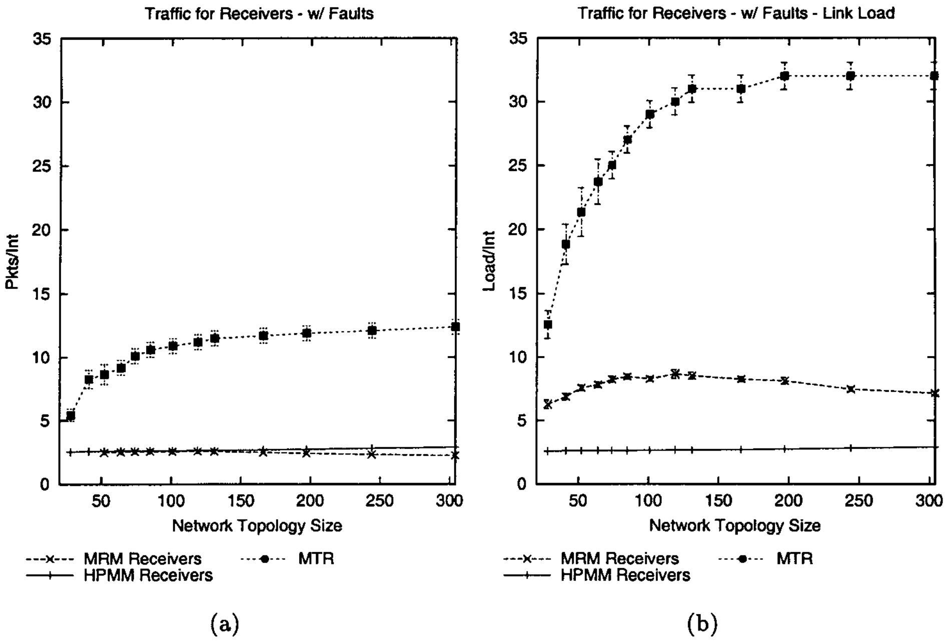


Figure 3: Average Receiver Traffic with Fault Occurrences

performs best in both cases. HPMM signaling scales best with a growing network topology and large deployment of agents.

Because HPMM defines a hierarchy of agents, it limits communication to local area. It is obvious that most messages in HPMM travel only short distances, in general only to the next hop (we assumed a daemon is deployed at each hop). Active testing performs less well, as can be seen in Figure 4, which graphs the distances in router hops traveled on average by packets in each protocol. The depth of the simulated multicast tree topology, which is the maximum distance any packet can travel, is also graphed for comparison.

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 Figure 5, we show network traffic for receivers only; without faults, no reports are send to the management station. The MTR approach result in the most traffic overhead of all schemes, since multiple subcasts are conducted. Passive testing requires no additional traffic other than keep-alive messages. Active test traffic still remains less than MTR subcasts

7. 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 are not able to 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-SSM, 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 management based on active-testing and non-aggregated fault reporting, as well as other schemes. Furthermore, unlike recently proposed approaches, we have shown HPMM can achieve better performance even without modifications to multicast routing. We also discussed fault types and fault correlation in the network and how they can be efficiently detected with HPMM.

Since HPMM requires software daemons to be deployed in the network, it is restricted to intra-domain usage. We have presented possible ways to use HPMM over multiple domains within inter-domain troubleshooting protocols such as GDT. Similarly, we wish to extend HPMM to work between hosts and proxies across 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 be to examine extensions to the basic mf-similarity scheme we have introduced in this paper.

8. ACKNOWLEDGMENTS

The authors would like to thank Kevin Almeroth of UC

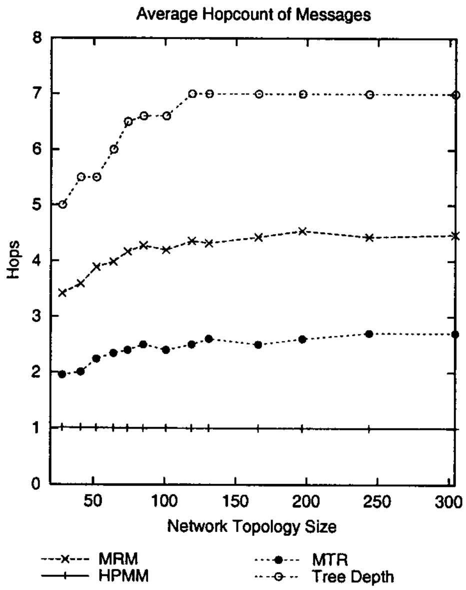
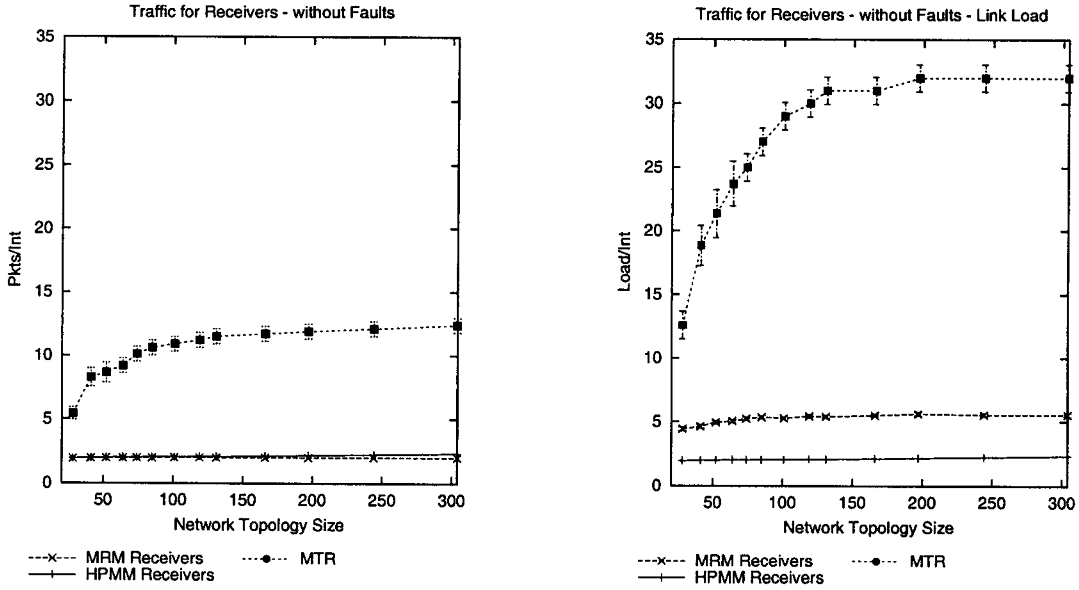


Figure 4: Average Message Hopcount



(a)

(b)

Figure 5: Average Receiver Traffic without Fault Occurrences

Santa Barbara, Supratik Bhattacharyya of Sprint Labs, and Ed Perry of Hewlett Packard for their insightful comments on this work.

9. 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 *ACM Multimedia '96*, pages 437–438, Nov. 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. V. 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, draft-ietf-rmt-bb-tree-config-*.txt*, 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] M. Inc. Mview utility. Internet Web Page. http://www.merit.edu/ mbone/mviewdoc/Welcome.html.

[27] H.-P. 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, Sept. 12–16 1998.

[29] F. LoPresti, N. G. Duffield, J. Horowitz, and D. Towsley. Multicast-Based Inference of Network-Internal Delay Distributions. Technical Report UM-CS-1999-055, University of Massachusetts, Amherst, Computer Science, Nov., 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] S. McCanne and V. 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] D. Thaler. Globally-distributed troubleshooting (gdt): Protocol specification. *IETF Internet Draft*, September 1997. draft-thaler-gdt-spec-*.ps.

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

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

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

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