Supporting Multicast Deployment Efforts: A Survey of Tools for Multicast Monitoring

Kamil Saraç and *Kevin C. Almeroth*
Department of Computer Science
University of California
Santa Barbara, California 93106-5110
{ksarac, almeroth}@cs.ucsb.edu

October 2000

Abstract

As the Internet is expected to better support multimedia applications, new services will need to be deployed. An example of one of these next-generation services is multicast communication, the one-to-many delivery of data. Over the last ten years, multicast research as well as deployment efforts have both been major areas of interest. In order to bridge the gap between the initial deployment experiments and the availability of multicast as a robust network service, there needs to be a full complement of multicast monitoring tools. In this paper we first survey the debugging, modeling, and management tools that have evolved along side the Internet's multicast infrastructure. Through this survey, we have observed important generalizations in three areas: (1) the challenges unique to monitoring multicast, (2) a methodology common to many multicast monitoring tools/systems, and (3) a set of considerations important to the development of new tools/systems. Using these generalizations we present two of our efforts to evaluate multicast reachability in the Internet. We also use these generalizations to evaluate some of the more recent efforts to develop large-scale management platforms.

1 Introduction

Traffic generated by multimedia-based applications have evolved into a significant portion of Internet traffic[1]. As a result, there is a need to develop better mechanisms to support multimedia traffic delivery. Initially applications used the reliable file transfer service provided by the Transmission Control Protocol (TCP). In addition to this service, one class of more recent multimedia-based applications are delivering audio and video in real-time to potentially millions of receivers using a multicast-capable version of the User Datagram Protocol (UDP).

Use of UDP is driven by the lack of a suitable transport-layer alternative. TCP is not an option with its heavyweight logical connections, congestion control, and in-order style of reliable transfer. By using UDP, critical services like congestion control, reliability, etc. need to be handled by the application. Furthermore, neither congestion control nor reliability are easy problems to solve as the semantics of these two services change in the face of the unique requirements of multimedia traffic. Reliability is often less critical than real-time delivery. Congestion control is more difficult because there is no feedback loop and because semantics are closely tied to TCP's method of using acknowledgments. Beyond these issues, challenges are created because of the need to improve the Internet's ability to handle real-time multimedia traffic. New network-services like quality-of-service, security, multicast delivery, and in-the-network processing have all been proposed as solutions.

Of particular interest is multicast communication. Multicast offers new opportunities to reach tens, thousands, even millions of receivers. The fundamental service offered by multicast is to solve the bandwidth bottleneck problem at the content server. Multicast allows one copy of each packet to be sent from a source. The network then replicates it at key branching points along a tree connecting all interested receivers. However, with the first bottleneck solved, others present themselves. One is solving the problem of being able to monitor the quality and reachability of potential receivers. The broader challenge is to effectively and efficiently monitor and manage multicast traffic, groups, and participants.

Providing monitoring and management support for multicast is an important component of building a robust Internet service. In this paper, we focus on monitoring and take the approach that management, debugging, and modeling activities are all based on the kinds of data gathered from monitoring efforts. "Management" is the process of making intelligent decisions on how to provision networks and services; "debugging" is narrowly focused on solving problems immediately affecting service; and "modeling" is the process of understanding temporal and spatial characteristics. We believe these activities are critical to the successful deployment of a robust multicast service.

In this paper, we start by briefly describing the key multicast monitoring tools that have been developed since multicast was first deployed in the Internet. The effort to classify and describe the various tools gives us insight into how these tools have evolved, why they have evolved, their weaknesses, and their strengths. To this end, we have developed a generic architecture and a set of metrics to compare and contrast various monitoring systems. Both of these are useful in understanding how tools relate to one another. As a way of exercising our architecture and metrics, we present a case study focused on multicast reachability monitoring. The case starts by examining *sdr-monitor*, a basic, easily deployable system that depended on existing MBone application semantics. Lessons from *sdr-monitor* motivated the design of a more general protocol called the Multicast Reachability Monitor (MRM) protocol. We describe these two systems in detail, identifying the weaknesses of the first, and discussing how these weaknesses have been addressed by the second. Finally, we also describe several other platforms useful for multicast monitoring, again with the goal of assessing their strengths and weaknesses.

The remainder of this paper is organized as follows. Section 2 catalogs the set of early multicast monitoring tools. In Section 3, we generalize the challenges and mechanisms related to multicast monitoring. Section 4 is a case study specifically focused on reachability monitoring. Section 5 evaluates a number of additional monitoring systems. The paper is concluded in Section 6.

2 Early Multicast Monitoring Tools

In this section, we look at the early monitoring tools developed for various multicast network monitoring tasks. These tools are primarily used for three closely related purposes: management, debugging and modeling. Figure 1 shows a distribution of multicast monitoring tools based on this classification. In this figure, the classes do not have strict boundaries between them—a monitoring tool can belong to more than one class. Time progresses from top to bottom, starting with the initial deployment of the Multicast Backbone (MBone) in 1992. 1997 marks the beginning of efforts to deploy hierarchical multicast routing[2]. The relative proximity among the tools are based on their functionality (horizontal proximity) and their introduction time (vertical proximity). The solid lines show where one tool has significantly influenced another. These kind of relationships exist because the later tool either (a) used the information produced by the earlier tool or (b) used a collection technique similar to its predecessor. Finally, at the bottom of the figure, we show a cloud including the potential candidates for next generation multicast monitoring systems.

Another useful characteristic for differentiating among the tools is how they perform their monitoring. Most of the tools in this figure perform *passive* monitoring. This means that the tools simply observe data as it already exists in the network. On the other hand, there are a number of *active* monitoring tools (indicated with a '*') that can create and use test data to perform their monitoring functions. In the rest of this section, we briefly describe the tools in Figure 1.

2.1 Debugging Tools

Soon after the deployment of the MBone, there was a need to understand how well the infrastructure was doing and where problems were occurring. As a result, the first set of monitoring tools developed were debugging tools. These tools were developed by researchers actively working on deployment and maintenance of the early MBone. They had narrow focused functionality and required an in-depth understanding about the operation of multicast routing protocols, state of the

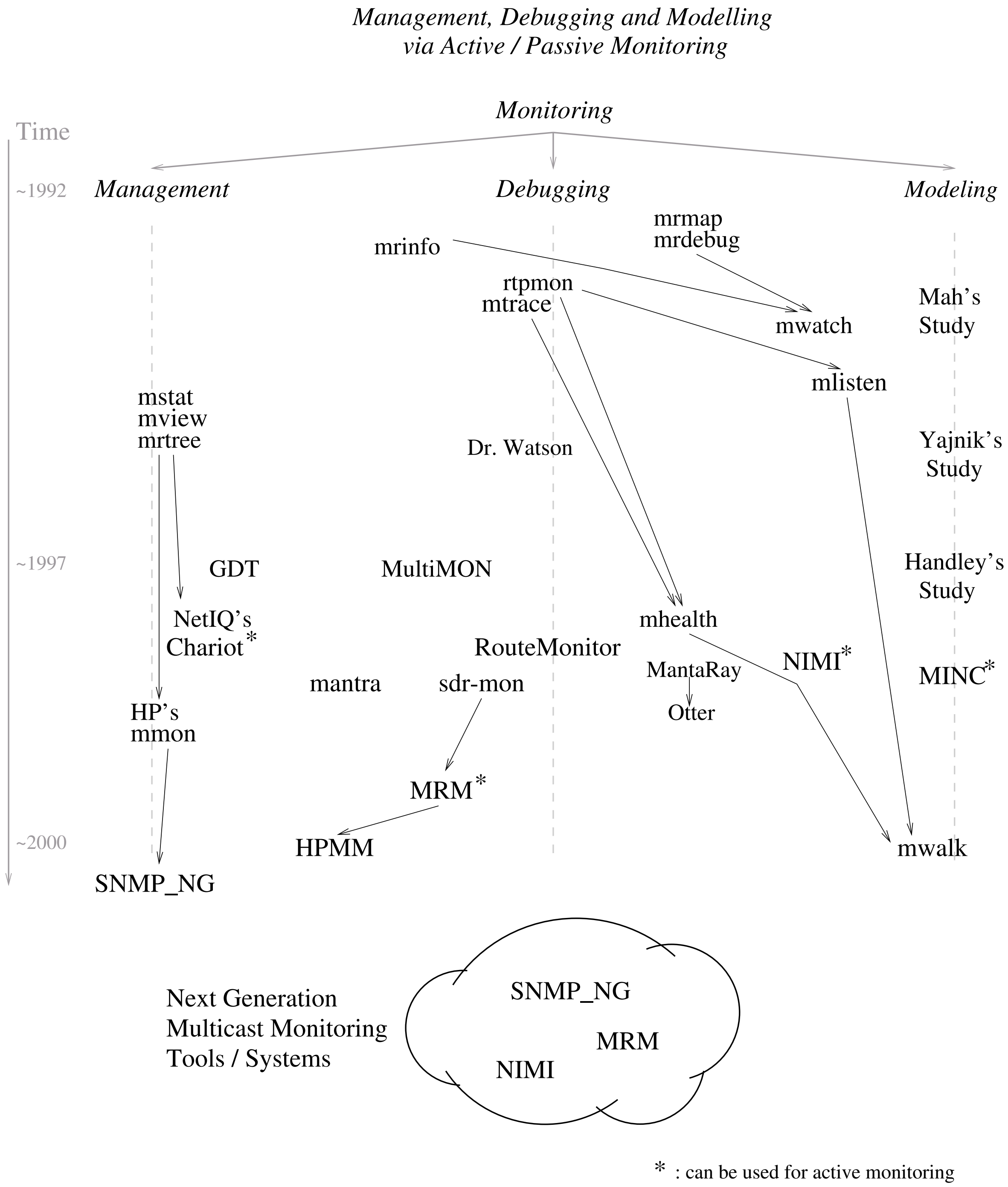

Figure 1: Classification of multicast monitoring tools.

MBone topology, and an ability to diagnose problems based on tool feedback. The tools included in this category include the following:

1. **Mrinfo**: The *mrinfo* tool reports on the tunnels and multicast-enabled interfaces for a router or end-host running multicast routing code[3].

2. **Mtrace**: The *mtrace* tool is used to return a snapshot of the set of links used to connect a particular source with a particular destination[4]. Additional *mtrace* options allow a user to measure the number of multicast packets flowing across each hop. The *mtrace* tool is one of the best ways of discovering the flow of multicast packets through a network.

3. **Rtpmon**: The *rtpmon* tool is one of the more useful tools for monitoring active multicast groups[5]. *Rtpmon* joins a particular multicast group address and receives feedback reports from all receivers. These feedback reports are generated by group members using the Real-time Transport Control Protocol (RTCP) which is part of the Real-time Transport Protocol (RTP)[6].

4. **Mhealth**: The *mhealth* tool combines *rtpmon* and *mtrace* and displays a real-time, graphical representation of a particular group's multicast tree including loss information[7, 8].

5. **Dr. Watson**: This tool allows a person debugging configuration problems to send, receive, and monitor packets of various protocol types[9]. Of relevance to multicast is the ability to test the Internet Group Management Protocol (IGMP)[10].

6. **MultiMON**: *MultiMON*, the IPmulticast Monitor, is a tool that collects, organizes, and displays statistics about multicast packets flowing across a Local Area Network (LAN)[11].

7. **Route Monitor**: The *Route Monitor* tool is used to collect routing protocol messages for the Distance Vector Multicast Routing Protocol (DVMRP). This information has been used to identify a number of protocol implementation bugs[12].

The original MBone was a virtual flat network on top of the Internet. The rapid growth of this virtual topology eventually began to suffer instability and scalability problems. Starting in 1997, this topology has transitioned to a hierarchical structure. This transition has spurred the development of a new set of monitoring tasks: reachability monitoring among multicast enabled domains and monitoring the functioning of the inter-domain multicast routing protocols. The following are tools created to handle these new tasks:

8. **Sdr-monitor**: *Sdr-monitor* is a tool developed to monitor inter-domain multicast reachability[13, 14]. It is discussed in detail in Section 4.2.

9. **Mantra**: *Mantra* collects multicast routing table information from a number of Internet backbone routers and parses this information to create a global view of various statistics about the topology[15].

10. **MantaRay, Otter**: Both these tools, developed by the Cooperative Association for Internet Data Analysis (CAIDA), are visualization tools. *MantaRay* is an interactive visualization tool for the MBone topology[16], and *Otter* is a general-purpose topology visualization tool[17]. *Otter* is used in systems like *Mantra*.

11. **MRM**: The Multicast Reachability Monitor (MRM) protocol is a developing protocol which aims to provide support for both intra- and inter-domain multicast monitoring tasks[18, 19]. MRM is discussed in detail in Section 4.3.

12. **HPMM**: The Hierarchical Passive Multicast Monitor (HPMM) system provides MRM-style monitoring but implements more effective reporting mechanisms to avoid implosion[20].

2.2 Management Tools

The term management has become synonymous with the Simple Network Management Protocol (SNMP)[21] and its Management Information Bases (MIBs)[22]. However, most of the early management activities conducted in the MBone were done without the assistance of SNMP. The reason for this is a side effect of the way the MBone evolved. People working on the MBone were more focused on establishing basic functionality than developing robust management tools. Instead of developing standards for management information and then the tools to utilize this information, efforts were focused on quickly creating effective tools. Developing SNMP-based tools is time consuming and it takes considerable time to standardize and implement MIBs. Solutions were needed quickly and SNMP was not typically considered as an option. However, this trend is changing. There has been a demand for more SNMP-based tools and some are becoming available. The set of management tools developed to date include the following:

13. **Mstat, mview, and mrtree**: These three tools were developed at Merit Networks as prototypes for SNMP-based multicast management[23]. *Mstat* is used to access multicast-related MIBs. The collected information is presented as text-based tables that can then be used by other tools. For example, *mview* is a tool that uses *mstat* to collect information about multicast sessions. *Mview* then displays this information graphically. *Mrtree* uses cascaded SNMP router queries to provide a text-based representation of a particular multicast group's topology.

14. **GDT**: The Global Distributed Troubleshooting (GDT) system provides mechanisms to detect and report network problems across administrative domains[24, 25]. It is discussed in more detail in Section 5.

15. **Chariot**: Chariot, originally developed by Ganymede Software, is a network performance test tool that allows distributed, end-to-end performance tests anywhere in a network. It provides a highly flexible way to test a wide variety of local and wide area networking environments and infrastructures.

16. **Mmon**: *Mmon* is an SNMP-based commercial tool prototype developed at Hewlett-Packard Labs[26]. The prototype automatically discovers and monitors the status of IP multicast routers and topology. From a visual map, an operator can see the state of the multicast infrastructure and query statistics about traffic activity.

17. **SNMP_NG**: The next generation of SNMP is designed to have better security, configuration capabilities, and scalability[27]. SNMP_NG is discussed in more detail in Section 5.

2.3 Modeling Tools

Another use of monitoring tools is modeling various properties about network characteristics and user trends. The collected information helps in understanding protocol behaviors, user trends and multicast network characteristics. This information then can be used to refine protocol specifications, fine tune network configuration parameters, and derive link-level network performance statistics. Modeling requires collecting long-term data and processing it to produce statistical results for various characteristics about the network. The following tools and studies fit into this category:

18. **Mrmap, mrdebug**: These two tools were developed early in the history of the MBone to help map and then debug the MBone[28]. While they are considered debugging tools, the ability to show the multicast topology also makes them modeling tools.

19. **Mwatch**: The *mwatch* tool was another early tool to build a view of the MBone topology[29]. It worked by "walking" multicast-capable routers using the *mrinfo* tool.

20. **Mah's Study**: Mah conducted one of the first studies on multicast traffic analysis[30]. The study focused on the volume and types of traffic transiting Berkeley's tunnel to the MBone.

21. **Mlisten**: *Mlisten* is one of the early monitoring tools that collected group membership information[31]. Information collected using *mlisten* has also been used to model long-term multicast usage trends[32].

22. **Yajnik's Study**: Yajnik, et al. have characterized a controlled MBone session by analyzing packet loss statistics from 11 participating sites[33]. Their work used this data to examine the spatial and temporal correlation between packet loss and links in the tree.

23. **Handley's Study**: Handley has developed tools to log RTP/RTCP packets and collect *mtraces* for an MBone session[34]. The data sets have been used to manually create a picture of the multicast tree and collect various statistics about links in the tree.

24. **MINC**: The Multicast-based Inference of Network-internal Characteristics (MINC) project identifies internal network performance characteristics based on end-to-end multicast measurements[35]. MINC uses RTCP based data reception reports to infer link-level loss rates and delay statistics by exploiting the inherent correlation in performance observed by multicast receivers.

25. **NIMI**: The National Internet Measurement Infrastructure (NIMI) provides an architecture in which a collection of measurement probes cooperate to measure various properties of Internet paths and clouds[36]. NIMI is discussed in more detail in Section 5.

26. **Mwalk**: The *mwalk* tool is used to model multicast tree topology characteristics using archived *mtrace* and *mlisten* data[37]. Multicast trees are characterized according to depth, degree, and degree-at-depth metrics for a variety of generated and real data sets.

2.4 Limitations of Existing Tools

Early multicast monitoring tools mainly focused on debugging problems in the MBone. These tools were developed by people closely involved in the ongoing deployment and management of the

MBone. In addition, the day-to-day management of the MBone was a relatively unique effort in that these functions were handled via a mailing list. Discussions were typically informal with the main responsibilities distributed among a few dedicated individuals. As a result, the debugging strategies and the tools in use have been influenced significantly by the requirements of dealing with the day-to-day problems that arose as the MBone grew. This evolutionary process has created several problems:

- **Dependence on Application Layer Data:** Many early monitoring tools are dependent on all group members using an application-layer protocol to exchange participant data. Examples of monitoring tools that rely on application-layer protocols include those that use RTCP, e.g. *rtpmon*, *mlisten*, and *mhealth*. Another example is *sdr-monitor* which uses session announcements sent using the Session Announcement Protocol (SAP)[38]. Dependencies on these kinds of protocols create potential monitoring problems. In particular, difficulties in accurately monitoring group statistics occur when not all decoding/display tools properly implement the application-layer protocol. The accuracy of monitoring efforts is complicated when group members have the ability to receive data without sending feedback information.

- **Requiring Significant Multicast Expertise:** Most of the tools developed during the initial deployment of the MBone required an in-depth understanding of how multicast works. The problem with this requirement is that a high proficiency in multicast operation is a difficult skill to find in most Network Operation Center (NOC) personnel.

- **Lack of SNMP-based Tools:** Few of the early tools used SNMP as the mechanism to gather data or monitor networks. Furthermore, there was little effort to develop multicast-related MIBs, and even lesser effort to implement these in critical places in the network. As a result, the gap between the techniques used by the multicast experts and the needs of those expected to manage multicast networks has become even larger.

- **Lack of Commercial Tools:** There are only a few commercially available multicast monitoring tools. NOC personnel, who are expected to keep multicast running properly, cannot rely on tools created as proof-of-concept prototypes. Since IP multicast has been in active development, companies have not been willing to put in effort to develop high-quality tools. This trend is changing. Figure 1 shows there have been some recent commercial interest in developing management tools.

3 Generalizing the Task of Multicast Monitoring

In the previous section, we discussed a number of early multicast monitoring tools. This discussion gives us enough insight to attempt developing a set of general characteristics for multicast monitoring systems. We start with a discussion of why multicast monitoring is difficult and different from unicast monitoring. Then we present a generic architecture and a set of general metrics useful for categorizing and evaluating multicast monitoring tools/systems.

3.1 The Challenges of Multicast Monitoring

Monitoring multicast traffic is somewhat similar to monitoring unicast traffic, but there are differences. The key difference derives from the simple fact that multicast traffic can be destined for multiple receivers. With multicast, this level of abstraction carries additional importance because of the added complexity associated with delivering a packet to multiple receivers. Instead of monitoring connectivity between pairs of users, multicast deals with potentially very large groups of users. And instead of monitoring the links along a single path, multicast deals with links organized into a tree.

Anonymity of group members and use of UDP makes it difficult to monitor multicast groups. For example, the current multicast model is an open service model that supports sessions in which anyone can send data to a multicast group and/or join and receive data from the group[10]. In this model, senders and receivers may not be known to each other. Support for dynamic groups makes multicast management more difficult. In particular, reachability monitoring–the task of verifying if multicast data from a session source can be received at a session receiver site–requires additional mechanisms. This is because in the current IP multicast service model there is no implicit group coordination or management. Therefore there can be no implicit way of knowing the group members.

As an example of why the specific characteristics of multicast are more of a challenge than unicast, we consider the case of monitoring reachability. One mechanism for determining who group members are and whether there exists reachability between source(s) and receiver(s) is the *ping* utility. In unicast, ping allows a source/receiver to test bi-directional reachability to a peer receiver/source. In the case of multicast, because of the open service model and because ping requests are made to a group instead of a receiver, the source does not know from whom and from how many group members to expect responses. This creates a number of problems. First, there is the problem of *implosion* which can occur if a very large number of group members choose to send a response within a small interval. Second, the responses that are sent may only be from a subset of group members. Receivers who do not have bi-directional connectivity with the source will not be heard, i.e. receivers who do not hear the ping request (in the case of a broken link), or receivers who do not have connectivity in the reverse direction. On the other hand, a multicast version of ping tool that is truly analogous to the unicast ping should return reachability status for all the receivers in the group. Figures 2 and 3 compares the behavior of multicast ping as it currently exists and an ideal version of how it should exist.

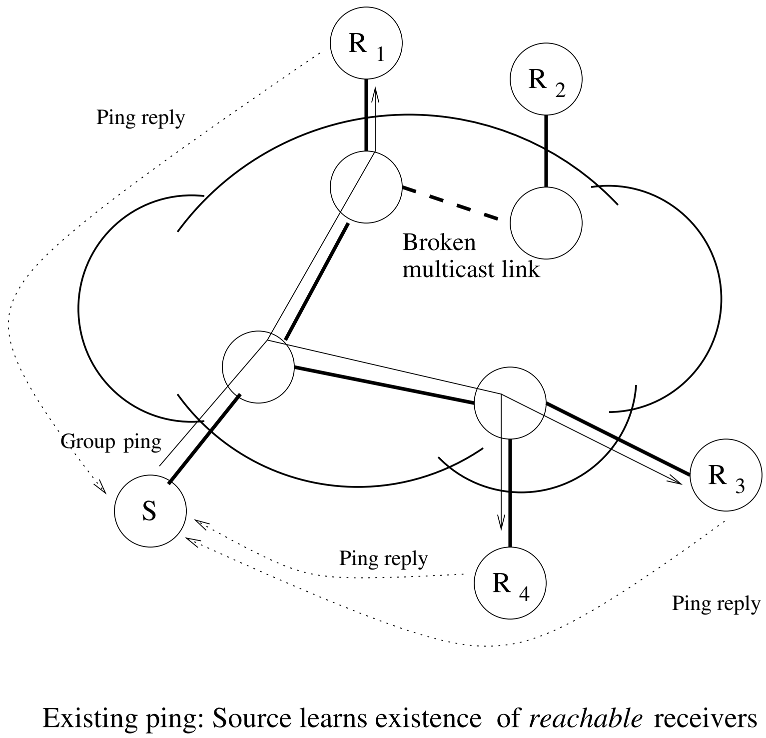

Figure 2: Semantics of the current multicast ping.

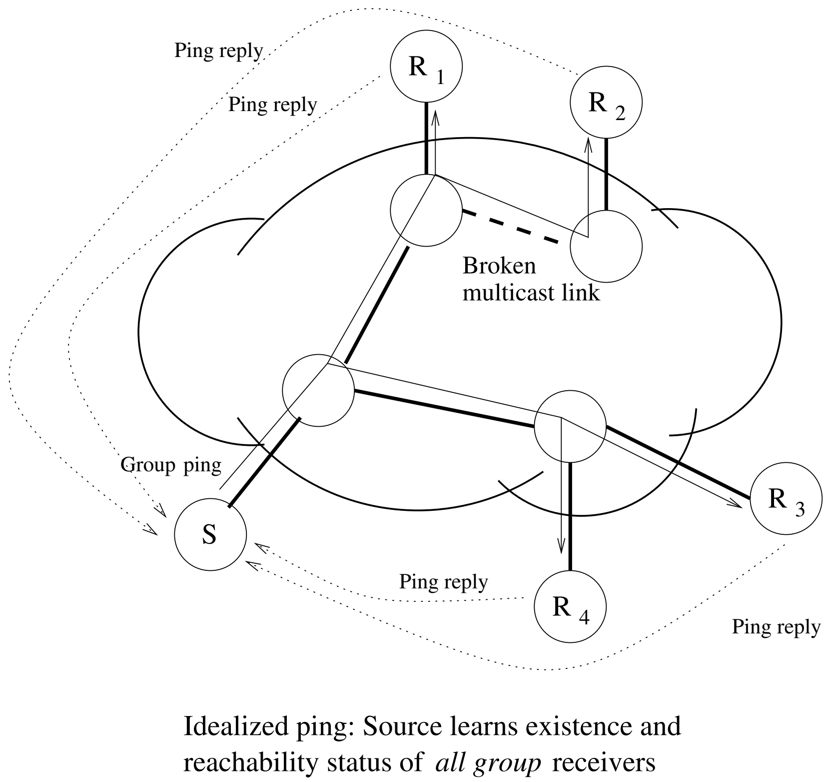

Figure 3: Semantics of an idealized multicast ping.

3.2 A Generic Architecture for Multicast Monitoring Systems

Based on the evolution of multicast monitoring tools, we have developed a general architecture for multicast monitoring systems. In this architecture we focus on the basic system components and their functionality. In general, a multicast monitoring system has a *manager* component and one or more *agent* components. Figure 4 presents an example of a basic system architecture for multicast monitoring. This figure includes two different monitoring scenarios: one using network devices for intra-domain monitoring, and one using end-host systems for inter-domain monitoring. The functionality in these scenarios can be divided into three steps:

1. **Manager configures agents for a particular monitoring task**: In this step, the manager site sends the necessary configuration parameters to the agents who will perform the monitoring task. The type of communication between the manager and agents is an important consideration. Depending on the scale and complexity of the system, this communication may be as simple as a UDP-based message exchange (with or without reliability) or it may

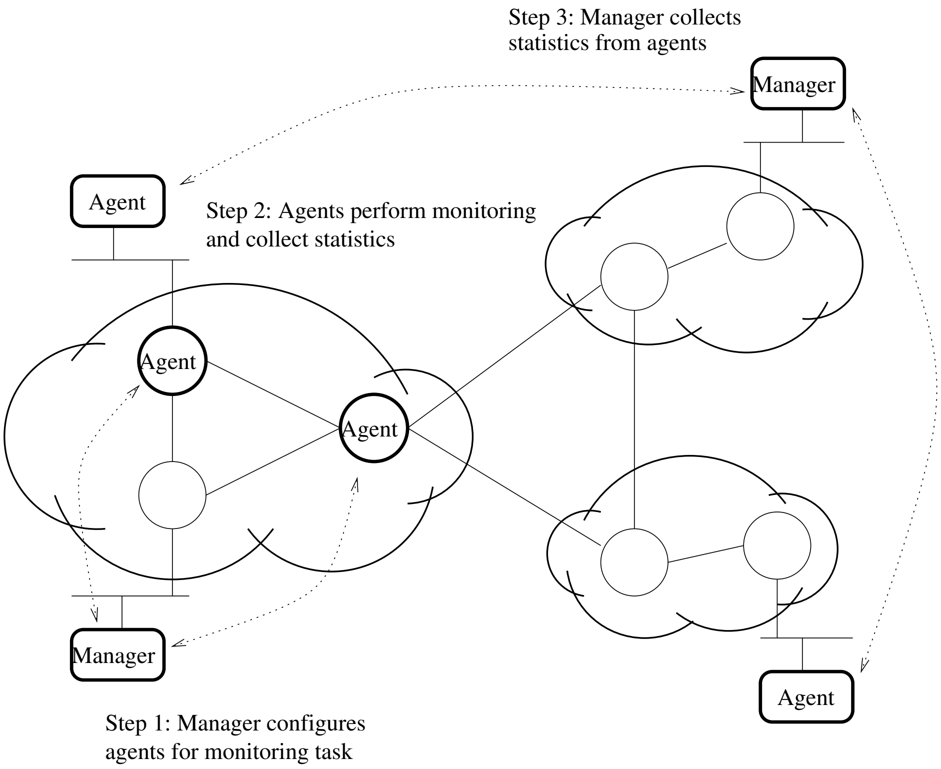

Figure 4: A generic monitoring system architecture.

be as complex as some type of authenticated and encrypted message exchange. The use of multicast is sometimes an option but this solution usually creates as many problems, e.g. reliability and implosion, as it solves.

2. **Agents perform monitoring task**: After the agents are configured for a specific monitoring task, they each conduct their measurements. The scope of agent functionality in this step can vary widely. On one end of the spectrum, agents can simply send state information to the manager based on thresholds or simple probabilities. More complex monitoring functions may involve active sourcing of test data, generating statistical data based on a received packet stream, or aggregating results received from other agents.

3. **Manager collects and processes agent reports**: Like the other steps, the report/result collection mechanism in monitoring systems has significant variability. In some cases, a session manager can poll agents to receive reports. In other cases, agents send their reports directly to the manager. As mentioned above, some systems also use agents as intermediary points—using them to form a reporting hierarchy, thus avoiding report implosion.

Developing this generic architecture is useful in understanding the functional mechanisms of multicast monitoring systems. We believe that this architecture captures the essential components of most monitoring tools/systems. However, it should be considered as only a first approximation. In general, the components and their functionality depend highly on the requirements of the particular monitoring task. For example, a monitoring task that requires observing multicast traffic flow on a particular LAN would require only a simple version of this architecture. The architectural components may even be combined into one relatively straightforward tool, e.g. *MultiMON*. On the other hand, a system's architecture may be fairly complex. A monitoring system may have to generate test packets from numerous locations in a coordinated attempt to capture an accurate snapshot of network-wide reachability. In this case, the manager would have to configure a number of monitoring daemons in the network to collect and report statistical data. Using the results, the manager might then take action based on the interpretation of these reports, and then correct observed problems automatically. In the next section, we follow up the generic architecture with a set of characteristics useful for further describing multicast monitoring systems.

3.3 Characteristics of Multicast Monitoring Systems

After studying the architectural properties of multicast monitoring systems, we now turn our attention to the operational characteristics of these systems. In this section, we present a number of characteristics that should be considered when designing and developing multicast monitoring systems. Furthermore, these characteristics can be used as *metrics* to compare the effectiveness of any of the multicast monitoring systems we have discussed. Later in this paper, we use these metrics to transition from a discussion of tools that already exist to evaluating tools/systems under development. The important characteristics that should be considered when designing or evaluating a multicast monitoring system include:

- **Intra- vs Inter-Domain Support**: A multicast monitoring system may need to support both intra-domain and inter-domain monitoring. Each kind of monitoring has potentially different functional requirements. For example, access to network devices in the intra-domain is typically unlimited and devices can be fully controlled by the NOC. NOC personnel are expected not only to monitor the network but also to solve problems that arise. Monitoring in the inter-domain is based on much more limited access to devices. Typically, instead of monitoring with the goal of management, the goal is instead to confirm that (a) problems do exist, and (b) problems are located in a remote domain.

- **Scalability**: Scalability requires architecture bottlenecks (e.g. complexity, message overhead, processing, etc.) to increase sub-linearly with (a) the number of monitoring devices, and (b) the overall size of the network. A multicast monitoring system should adopt mechanisms to prevent unnecessary traffic on the network and excessive processing load on the participating nodes. One particularly important scalability requirement is the ability to control report implosion. Implosion can occur when a centralized collection site is used and the system supports *alarms*, i.e. asynchronous reports of error conditions. For example, while monitoring availability and reception quality of multicast data in a multicast session, failure of a critical link close to the session source would likely cause many receivers to generate alarms.

- **Security**: Security is another important issue in monitoring systems. Network devices used in intra-domain multicast monitoring tests are usually production components and should only be accessed in a controlled manner by authorized personnel. These devices should be configured to accept connections and/or service requests from management sites only. Communication between management sites and these devices should be encrypted and/or authenticated to prevent malicious attacks. For the inter-domain case, the potential for attack and mistrust is much greater. In addition to intra-domain concerns, inter-domain security mechanisms need to be used to control accesses by authorized users in remote domains.

- **Extensibility**: Extensibility deals with the issue of a system's ability to support the collection of new sets of data. SNMP offers a good model. In SNMP, there is a key distinction between the protocol to collect data and the kinds of data that can be collected. Similarly, multicast monitoring systems should support collection of as little or as much data as is necessary.

- **Device Flexibility**: A monitoring system that is able to support data collection at *internal* network devices as well as edge devices is likely to offer more useful monitoring data. Again, SNMP offers a good model to follow. SNMP support is provided in almost all types of devices from routers and switches to printers and peripherals. Edge devices provide a good end-to-end, user-level view, while internal network devices help show exactly where problems are occurring.

- **Multicast Independence**: In a multicast monitoring system, successful communication between a monitoring coordination site and monitoring agents, even in the presence of faults, is important. Monitoring systems should not depend solely on the availability of multicast for communicating control information. If multicast were used as the only mechanism to provide control information exchanges, monitoring and reporting mechanism breakdown and the effectiveness of monitoring jeopardized. However, multicast is important as it is a particularly effective mechanism for achieving scalability. Several of the tools discussed so far (e.g. *mtrace* and RTCP-based tools) provide excellent functionality but only when multicast is working properly.

- **Abstraction and Presentation**: An important step after monitoring data has been successfully collected is how to turn the data into useful information. Some multicast monitoring systems, like *sdr-monitor*, *MantaRay*, and *Mantra* have shown how to visualize monitoring data and create easy-to-decipher results.

4 Case Study: The Evolution of Multicast Reachability Monitoring

Reachability monitoring is an excellent example of a challenging multicast monitoring task. Since the transition of the multicast infrastructure into a hierarchical topology in 1997, monitoring reachability between Internet domains has become an important problem. In this section, we use reachability monitoring as an example of a typical monitoring task. Our objective is to discuss two current systems, *sdr-monitor* and MRM, as instantiations of the generic multicast monitoring architecture developed in Section 3.2, and then evaluate them using the metrics developed in Section 3.3.

4.1 The Challenges of Multicast Reachability Monitoring

Reachability monitoring is one of the most important yet one of the most difficult multicast monitoring tasks. From a management point-of-view, successfully deploying multicast requires the ability to build confidence that the network is working in a correct and predictable manner. This requires mechanisms to monitor and verify successful multicast data transmission within and between multicast-enabled domains. For a globally-scoped multicast application, a number of potential receivers may be located in other domains and the availability of data to these receivers may be affected by reachability. Network operators must have the ability to ensure multicast reachability to all potential receivers. The dynamic nature of multicast trees and anonymity of group receivers are important properties of the current multicast model that make it difficult to verify reachability to all potential session receivers. In the rest of this section, we first identify the requirements for multicast reachability monitoring and then discuss *sdr-monitor*, and its offshoot, MRM.

Using reachability monitoring for our case study is useful because it lends itself well to the metrics we have developed for characterizing multicast monitoring systems. Obviously the ability to provide both intra-domain and inter-domain monitoring is important. Considering the other factors, the ability to scale to a large number of monitoring sites is important. A reachability monitoring system should also have some measure of security, should be able to determine reachability based on a variety of data, and should be adaptable to observation points in and at the edge of the network. Finally, the data collection mechanism should not rely only on multicast as a response mechanism, and the results should be displayed in an intuitive manner. With these considerations in mind, we now describe the *sdr-monitor* system.

4.2 *Sdr-monitor*

Sdr-monitor is an application-layer multicast reachability monitoring tool. It is based on monitoring multicast *session announcements* that are transmitted among multicast users. These announcements are widely used to convey information about active groups to potential receivers. The session directory tool (*sdr*) is a distributed tool used by researchers around the world to announce the availability of multicast audio, video, whiteboard, and/or text sessions[39]. In the *sdr-monitor* system, these periodic announcements are also used as heartbeat messages to monitor reachability between domains. *Sdr-monitor* participants listen to *sdr* announcements, and periodically report which announcements are seen at their site. *Sdr-monitor* then processes these reports and builds a real-time web page displaying a reachability matrix for the global multicast infrastructure. *Sdr-monitor* also archives the collected information for long-term analysis. The web site has become a useful monitoring and debugging tool for the multicast community. In addition, using archived data, an analysis of global reachability patterns was conducted. In the remainder of this section, we present the *sdr-monitor* architecture, provide some of the results obtained using *sdr-monitor*, and discuss the limitations of the tool.

4.2.1 *Sdr-monitor* Architecture

Sdr-monitor uses available session announcements from topologically and geographically distributed users to build a representation of the reachability status in the global multicast infrastructure. The *sdr-monitor* architecture, shown in Figure 5, includes the following components:

- **Session Announcement Originators:** Any user that sends multicast session announcements on the well-known session announcement address (using *sdr* or any other tool) becomes a source for *sdr-monitor* heartbeat messages.
- **Sdr-monitor Participants:** Any *sdr* user can become an *sdr-monitor* participant. The number of *sdr-monitor* participants is limited. The target number of active participants is approximately 25 to 30. Participants use a *sender script* to deliver their *sdr* cache entries to the *sdr-monitor* collection site (see Figure 5). The sender script is a small Tcl script that runs along with the *sdr* tool. When the *sdr* tool is started, it automatically invokes the sender script. When invoked, it forces *sdr* to write current set of announcements to the local disk and then sends these announcements to the *sdr-monitor* collection site via email. This process is repeated every hour. Using email is not particularly scalable, but provides a reliable method for collecting *sdr-monitor* participant reports.
- **Central Collection/Processing Site:** At the *sdr-monitor* collection site, a manager receivers emails from remote sites and processes them. The manager runs as a daemon process and periodically checks for incoming email messages. The manager uses these messages to generate a web page displaying a reachability matrix. The web page is updated continuously as new information is received. In addition, the manager takes a snapshot of the reachability matrix every hour and archives it for long-term analysis.

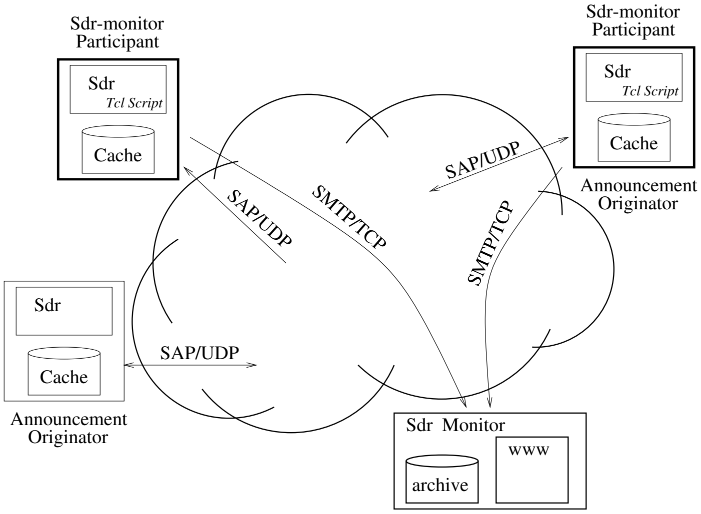

Figure 5: The *sdr-monitor* architecture.

4.2.2 *Sdr-monitor* Outputs, Results and Analysis

Sdr-monitor produces two outputs: a real-time web page and an archival data set. As mentioned above, the *sdr-monitor* web page displays the current view of global multicast reachability for all known global sessions for all *sdr-monitor* participants. By examining this real-time snapshot, the web page can be used to quickly identify reachability problems in the global multicast infrastructure.

The archival data set consists of the snapshots of the *sdr-monitor* web page taken once an hour. It is used for reachability analysis. This analysis is important in understanding long-term reachability characteristics of the global multicast infrastructure and in understanding the success of deployment. In the remainder of this section, we briefly summarize the key aspects of our analysis on a data set collected between April 1999 and September 2000. A more detailed analysis on a subset of this data is available elsewhere[13]. The analysis can be divided into four parts:

- **Step 1**: Data is processed to remove mis-formed and non-globally scoped *sdr* announcements. Session announcements with a Time-To-Live (TTL) value less than 127 are considered non-global session announcements and are removed from the data set. In addition, all administratively scoped session announcements are also filtered. Even though these announcements may have a global TTL (127 or larger), they will likely be blocked at administrative boundaries.

Lastly, all announcements that have not received a soft-state update in the previous hour are considered "stale" and are also filtered.

- **Step 2**: We have identified a number of artifacts of using *sdr*-based session announcements for monitoring reachability. These problems are mainly due to irregular participation behavior and irregular session-announcing-site behavior. Not all *sdr-monitor* participants run *sdr* continuously. This means that not all participants are continuously reporting their cached announcements. Since each participant has a potentially different picture of global reachability, their joining and leaving can cause dramatic changes in *sdr-monitor*'s results. Similarly the number of sites sourcing session announcements is also dynamic. Like participants who see different sets of sites, session-announcing-sites will be seen by a different set of participants. Each time a site starts or stops advertising a session, it affects the perceived global reachability. After identifying these problems, we further process the data to minimize their impact on the data set.

- **Step 3**: At this point, the remaining data is used to display the long-term reachability characteristics of the global multicast infrastructure. The results produced are based on calculating the *daily average reachability* for each session-announcing-site. This is computed by averaging the visibility of all *sdr-monitor* participants for each session-announcing-site. Visibility of a site is computed by dividing the number of participants receiving an announcement by the total number of active participants. For graphing purposes, we then divide session-announcing-sites into four groups based on their daily average visibility. The four groups are: 0%-25%, 26%-50%, 51%-75%, and 76%-100%. Figure 6 shows the breakdown of results over the 17 month collection period.

- **Step 4**: In the last part of our analysis, we attempted to identify the qualitative reasons for reachability problems. The main reasons that we believe cause reachability problems are (1) local connectivity problems at *sdr-monitor* participant sites, (2) inter-domain connectivity/peering problems, and (3) trans-oceanic connectivity problems[13].

4.2.3 Evaluation of *Sdr-monitor* as a Monitoring Tool

As a monitoring tool, *sdr-monitor* has a number of areas that could be improved. In large part, many of these problems relate to the use of SAP as a heartbeat mechanism. Furthermore, it is exactly this set of problems that MRM was targeted to solve. These problems include:

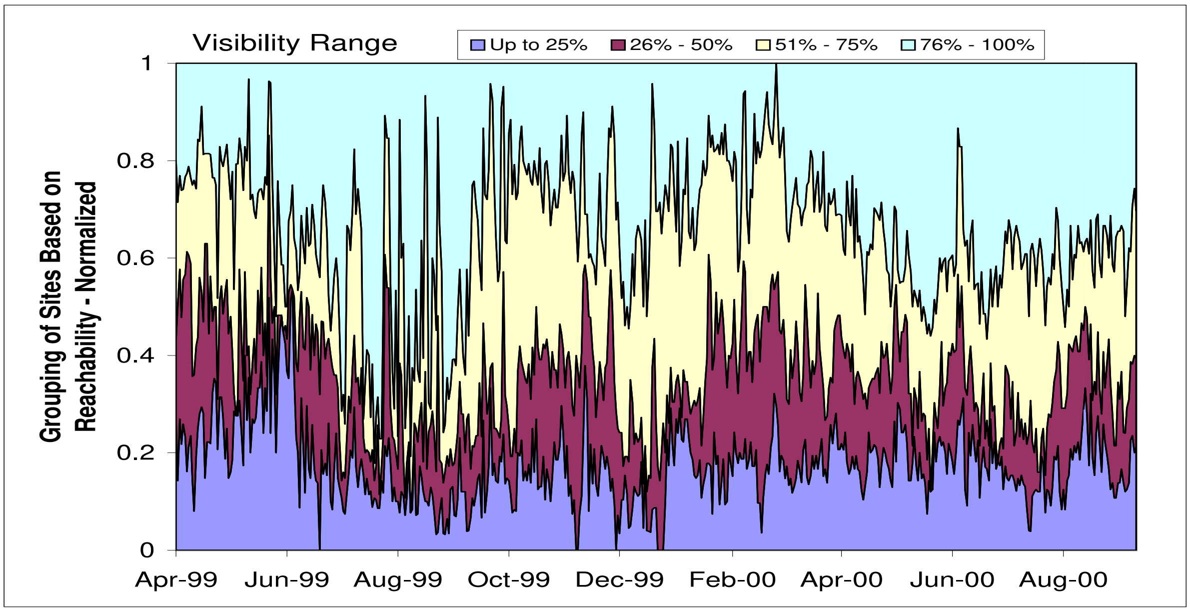

Figure 6: Average visibility for *sdr-monitor* session-announcing-sites.

- **Lack of flexible monitoring**: *Sdr-monitor* can only report reachability between sites that are advertising sessions and *sdr-monitor* participants. Furthermore, this reachability is in only one direction.

- **Lack of heartbeat message control**: *Sdr-monitor* cannot control the frequency of heartbeat messages sent by sources. Packets are sent periodically (approximately once every 5 minutes), and this may not be sufficient to establish the routing state necessary to measure reachability. Furthermore, periodic, single packet transmissions are not sufficient to give us a measure of the quality of the connections between sites.

- **Lack of consistent monitoring**: Because both sourcing sites and participants can come and go at will, the results can change dramatically even though overall reachability does not change significantly (see Figure 6).

Taking a broader look at *sdr-monitor*, its architecture has similarities to the generic architecture in Section 3.2. In the *sdr-monitor* architecture, the manager-to-agent interaction is not automated. The maintainers of *sdr-monitor* establish contact with potential participants and then send the *sender script* to them via email. The monitoring step is active only when participants are running *sdr*. The report collection step occurs using email back to the manager. Finally, using the criteria established in Section 3.3, we evaluate *sdr-monitor* as follows:

- **Intra- vs. Inter-Domain Support**: *Sdr-monitor* can provide support for monitoring on any scale as long as there are users willing to participate.
- **Scalability**: *Sdr-monitor* does not provide an explicit scalability mechanism. In the case of a large number of participants, the message processing load at the manager site may be significant. However, since reports are delivered at random periods, this helps avoid implosion.
- **Security**: *Sdr-monitor* does not have any explicit security mechanisms. It is therefore susceptible to malicious users submitting false reports or overwhelming the server by sending large numbers of bogus reports.
- **Extensibility**: *Sdr-monitor* depends on an existing application layer protocol mechanism and so has no real extensibility.
- **Device Flexibility**: *Sdr-monitor* is limited to end-host systems capable of running *sdr*.
- **Multicast Independence**: *Sdr-monitor* does not rely on multicast for communication between the manager and agents.
- **Abstraction and Presentation**: The *sdr-monitor* web page has a very useful display. The color coded matrix shows what a site should see and identifies the region for which reachability does and does not exist. We have observed that "clustering", both of reachable and unreachable sites, can be used to identify faults and even isolate their location.

4.3 The Multicast Reachability Monitor (MRM) Protocol

MRM is an under-development protocol to send and receive test multicast data streams and to collect information about the quality of the stream. MRM is designed to support basic reachability monitoring plus provide "hooks" to other management systems and tools. The development of MRM has been influenced by *sdr-monitor* and the need to create a better inter-domain reachability monitoring system. MRM is also being designed to be used as an intra-domain monitoring and management tool. As a result, MRM is truly a management system and a natural evolution of several monitoring tools.

The functionality provided by MRM is the ability to have a centralized management station configure test multicast sessions. Configurations include who the source(s) should be, what the transmission rate should be, who the receiver(s) should be, and criteria for reporting results. A network manager is responsible for coordinating the test sessions, processing results, and presenting them to a user. Generated traffic can be used to test basic reachability or test end-to-end capability. Tests can be conducted in advance of an event to confirm functionality or during an event to monitor quality. In addition to individual tests, a suite of tests can be conducted in which a frequently changing set of network devices is used. In this way, statistical testing can be performed for large networks.

The remainder of this section is dedicated to describing the MRM protocol in detail. This description includes an overview of MRM and its components, a list of MRM's message types, and an evaluation of MRM using the metrics developed in Section 3.3.

4.3.1 MRM Protocol Description

The primary goal of MRM is to provide network fault detection and isolation mechanisms for administrating a multicast-enabled infrastructure. An MRM-based fault monitoring system consists of two components: (1) MRM *agents* that source or sink test traffic, and (2) an MRM *manager* that configures tests, collects and presents fault information. The MRM protocol specifies the communication primitives used between MRM agents and the MRM manager. Lastly, it provides mechanisms to provide scalability, flexibility and security, etc. From this perspective, the MRM protocol can be considered a combination of a scalable version of SNMP's communication functionality and the flexible data structure of MIBs. A more detailed description of MRM agents and managers is as follows:

- **MRM Agents:** MRM agent code can potentially be run in any device in the network. The goal is to have routers implement limited agent functionality, and to have more capable end-hosts implement a richer feature set. Agents act as either *Test Senders* (TSs) or *Test Receivers* (TRs). A TS sources data packets in response to a request from an MRM manager. A test scenario may not require any TSs if the manager configures TRs to monitor traffic for an pre-existing, active multicast session. An MRM agent configured as a TR will receive traffic, collect statistics, and send reports. The specific actions taken depend on the capabilities of the agent and what the manager configures it to do.

- **MRM Manager:** An MRM manager initiates configuration requests to the MRM agents and assigns the roles of TSs and TRs. The MRM manager informs the TSs and TRs of the types of monitoring or diagnostic tests to run. The MRM manager also specifies the type of reports the TRs should send. Agents can be asked to send reports at specific intervals, only if certain thresholds are violated, or probabilistically. The MRM manager plays a key role in determining the usefulness of the tests as well as their impact on the network and the network devices. The MRM protocol specifies the basic interaction mechanisms between MRM managers and MRM agents. It also provides support for MRM managers to control and change the behavior of MRM agents during a test session. Functionality offered by an MRM manager can be as simple as a command line interface with a simple display of all responses, or it can be a sophisticated tool. A manager could be incorporated as part of an operational network monitoring tool, automatically deciding when to run test sessions; changing configuration of MRM agents during a test session; or deciding when and which types of reporting mechanisms to use.

An MRM test scenario has four basic steps. First, the MRM manager sets up the test. Second, the TSs send traffic to the TRs. Third, the TRs generate reports and send them to the MRM manager. And fourth, the MRM manager processes the reports. These four steps are very similar to the three steps shown in Figure 4. The second step in Figure 4—the data monitoring step—is broken into two steps for MRM. These four steps are shown in Figure 7 and described in more detail below.

1. An MRM manager instantiates a test scenario based on parameters from either a person or from a tool as part of a larger system. The MRM manager initiates configuration requests to the MRM agents and assigns the roles of TSs and TRs. The MRM manager informs the

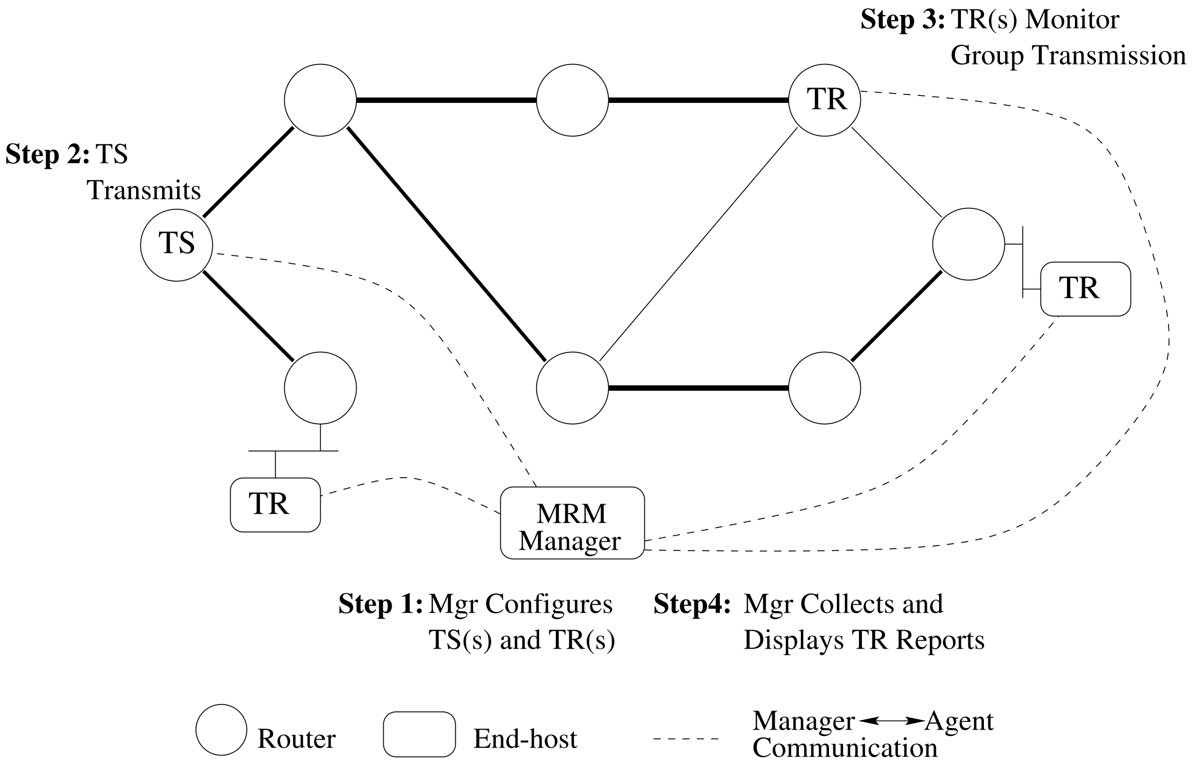

Figure 7: The architecture of an MRM system including message flow

TSs of the quantity and duration of traffic to generate and informs the TRs of the types of reports to generate.

2. TS(s) generate test traffic. In the case where an MRM scenario is monitoring real group traffic there may be no TSs. TRs will monitor whatever traffic they are configured to receive. One dependency between TSs and TRs is that the TRs must understand the transport and application layer packet headers used by TSs or the real traffic source. In the initial protocol design, the RTP packet header is used. This includes both traffic generation as well as TR status report messages. This allows re-use of existing RTP-based reception mechanisms and provides interoperability with existing RTP-based tools.

3. TRs generate *fault reports* and/or *status reports*. Fault reports are similar to SNMP alarms and are generated when a condition being monitored by the TR violates some threshold. For example, if loss exceeds a certain percentage, then the TR would send a report to the MRM manager. TRs may also send status reports but these are generated in response to explicit MRM manager requests. In this way, the MRM manager can periodically test the liveness of TRs.

4. The MRM manager receives and processes data from MRM agents. This function is not part of the protocol description but it is of critical importance nonetheless. We expect systems to be developed to take MRM reports and display the results. Or, we expect systems to be developed which format the MRM reports so that they can be passed to existing visualization tools[15, 17].

4.3.2 MRM Message Types

MRM functionality is based on the ability of an MRM manager to configure and interact with TSs and TRs. The four basic protocol message types include the following:

- **Test_Sender_Requests (TSRs)**: TSRs cause an MRM agent to begin sourcing (or stop sending if already sourcing) packets according to the parameters in the TSR packet, e.g. number of packets to send, inter-packet transmission delay, address of test session, type of packets to send, etc. TSR messages are sent using unicast UDP with acknowledgments. Additional *soft-state* updates may be carried in beacon messages (see below).
- **Test_Receiver_Requests (TRRs)**: A TRR message is delivered using the same method as TSRs (acknowledged UDP). A TRR can either be a request to the agent to become a TR or it can be a request to an existing TR to return a status report. In either case, a TRR packet includes the test session address, length of the test, kind of report to generate (RTCP or other), and when to generate a report (upon threshold violation, only upon request, or probabilistically).
- **Test_Receiver_Status_Reports (TRSRs)**: These reports are sent by the TRs to the MRM manager. The initial design is for status reports to use the RTP "receiver report (RR)" packet format[6]. In addition, MRM can be extended to support more detailed reports. Several extended report headers are currently under development[40].
- **MRM Beacon Messages**: One mechanism that MRM has that provides partial scalability is beacon messages. Beacon messages are sent periodically (recommended once every minute) by the MRM manager to a well-known multicast address (mrm.mcast.net). All TSs and TRs join this multicast group and listen. This beacon message contains a sequence number, the authentication data, the elapsed time since the last beacon message, and any active TSRs and TRRs for a particular scenario. The sequence number and elapsed time carried in a beacon

message can be used to verify MRM Manager liveness. This beacon mechanism has three purposes:

1. Allows TSs and TRs to learn the liveness of the MRM manager.
2. Allows the MRM manager to (unreliably) make large-scale changes to a test scenario. For example, an MRM manager can change the transmission rate for all sources, end a large-scale test prematurely, etc. This function is unreliable because MRM does not depend on the availability of multicast and therefore there is no way to guarantee that every target receiver can actually get the message.
3. Provides a soft-state re-assert mechanism in small-scale testing environments. A re-assert mechanism is useful in the case when network devices crash and then re-start. Beacon messages provide a way of letting these devices know that they are supposed to be participating in a test.

4.3.3 Evaluation of MRM as a Monitoring Tool

In Section 3.3 we listed a number of characteristics that can be used as metrics for evaluating multicast monitoring systems. In this section, we evaluate MRM based on these metrics.

- **Intra- vs. Inter-Domain Support**: MRM can be used both in the intra-domain and inter-domain. In the intra-domain, an MRM manager will likely have full access to all possible kinds of information. However, in the inter-domain there needs to be provisions to limit the kinds of tests that can be initiated from outside the domain. While this function is not a part of the MRM protocol specification, there are conventions in the protocol that allow an agent to *reject* a TSR or a TRR. By acknowledging the TSR or TRR but with a test duration of zero, the agent is effectively refusing to participate in the test. The conditions in which an agent can reject a test are subject to the guidelines imposed by each domain. Beyond the ability to control which tests are run, we expect inter-domain use of MRM to mainly be among end-host systems. Since end-host agents have access to less internal, proprietary network information, and a service provider cannot always control what a user does, it seems reasonable that end-host-based testing environments will be relatively easy to set up.
- **Scalability**: Currently, MRM has limited provisions for scalability. The problem is the desire to utilize the scalability of multicast but to avoid the dependence on multicast for robustness. The primary problem is that MRM has an asynchronous response mechanism.

Network engineers have the ability to configure test scenarios that can potentially generate overwhelming amounts of feedback traffic. This implosion problem exists and is addressed in a number of other scenarios, for example, in reliable multicast protocols[41]. Current research is underway to use some of these techniques in protocols like MRM[20]. Efforts are also underway to add multicast as a communication mechanism in more traditional protocols like SNMP[42]. The simple mechanism implemented by MRM is to utilize delayed feedback reporting. The MRM manager assigns a pre-determined report-delay to each TR. Each TR, upon detecting a fault, will randomly delay the sending of its report based on the report-delay period.

- **Security**: MRM provides necessary security mechanisms to protect MRM-capable network devices and end hosts from unauthorized use. Access rights to MRM agents are controlled using access lists, and MRM manager-to-agent communication uses the IP Security Authentication Header[43] with HMAC-MD5 transformation as the standard authentication algorithm[44]. As mentioned above, MRM agents also have the ability to reject TSRs or TRRs.

- **Extensibility**: MRM provides extensibility by offering multiple report types. Not only can MRM generate a wide variety of reports, but the kinds of reports generated can vary depending on the capabilities of the devices used in the test scenarios. Network devices are expected to perform only low-impact processing. On the other hand, end hosts can collect and store more detailed information about the arrival of *each* test session packet[40]. Currently, MRM's default report is the same format as RTCP Reception Reports. This allows other RTCP-capable tools to be able to receive and process MRM agent reports.

- **Device Flexibility**: MRM is designed to work in a variety of network devices and end-host systems, and has provisions to provide monitoring functionality based on the specific capabilities of the device.

- **Multicast Independence**: MRM uses beacon messages as an optional mechanism to provide scalability, but the basic protocol does not rely on it. Therefore, MRM operation is truly independent of multicast.

- **Abstraction and Presentation**: The protocol specification for MRM does not include a component for data processing and presentation. This functionality, which is expected to be included in the manager, is left to the discretion of the system implementor.

MRM provides much of the functionality that a good monitoring system should provide. However, the challenge is how to improve its scalability and then to ensure that there are commercial tools available that implement the protocol well and are useful in supporting deployment efforts. Another area that is still under development is MRM's interaction with SNMP. MRM's utility would be further improved if it had its own MIB and was able to provide the results of tests to SNMP-based tools.

5 Surveying Other Platforms for Multicast Monitoring

In the previous two sections, we examined *sdr-monitor* and MRM as two alternatives for reachability monitoring. In this section, we briefly discuss three more systems that are capable of monitoring multicast traffic and the health of the infrastructure: SNMP_NG, NIMI and GDT. The goal is to understand how these systems work, to compare and contrast their characteristics, and to understand what novel monitoring techniques each possesses.

SNMP_NG. SNMP is the common piece of many network management systems. Most NOC personnel are comfortable with SNMP and most are familiar with SNMP-based systems. However, SNMP is not without its deficiencies. Until recently, SNMP has been primarily used for monitoring and performance management within domains. If we were to take the current protocol version, SNMPv2, and evaluate it using our set of metrics, it would do poorly on inter-domain support, scalability, and security. To a large extent, these problems are being addressed by two threads of effort. First, there is effort in the Internet Engineering Task Force (IETF) to develop an SNMPv3 protocol with better support for security and device configuration[27]. Second, there are research efforts looking to improve SNMP's distributed management capabilities—primarily by adding scalability[42]. These efforts together are what we are calling SNMP-next-generation, (SNMP_NG). To this end, SNMP_NG has the following characteristics.

- **Intra- vs. Inter-Domain Support**: With new security features, SNMP should be much better at providing limited access to inter-domain peers. SNMP also continues to be an important intra-domain system.

- **Scalability**: With recent attention at incorporating multicast as a communication primitive, SNMP_NG should be much better at providing scalability.

- **Security**: A major focus of SNMPv3 is to improve security.

- **Extensibility**: SNMP continues to have significant extensibility through the flexibility of MIBs. Also, since SNMPv3 has the capability to configure devices, it will have the ability to create test scenarios and configure SNMP agents to source traffic.

- **Device Flexibility**: SNMP continues to offer significant device flexibility. SNMP capability is available in many devices ranging from switches and routers to printers and other peripherals.

- **Multicast Independence**: With efforts to add scalability through multicast, scalable versions of SNMP will likely be at least somewhat dependent on multicast. However, lessons learned in protocols like MRM should help avoid significant dependence.

- **Abstraction and Presentation**: There are already a number of SNMP-based management platforms that provide well-understood interfaces to NOC personnel.

NIMI. The National Internet Measurement Infrastructure (NIMI) provides an architecture in which a collection of measurement probes cooperate to measure various properties of Internet paths and clouds[36]. It has been developed based on the need for a global Internet measurement infrastructure. "NIMI platforms" installed at various end-hosts around the world are used for a variety of end-to-end network measurement projects. These platforms belong to network operators deploying them. Researchers that want to use these platforms for their measurements need proper access permissions. Access to NIMI platforms are controlled by authentication mechanisms. Access permissions are regulated by platform owners and these permissions are associated with a set of functions that can be performed on these platforms. Depending on the access permissions, one can upload new modules for performing specific measurements and use these platforms for measuring various network activities. Using the set of metrics from Section 3.3, NIMI performs as follows:

- **Intra- vs. Inter-Domain Support**: NIMI works well in both the intra- and inter-domains. The platform on which NIMI runs can be tightly controlled by the owner and access to sensitive data protected.

- **Scalability**: NIMI does not have specific mechanisms to provide real-time measurement scalability. In fact, NIMI seems to be more targeted at non-real-time measurements, i.e. tests that do not require results to be reported as fast as possible. In this sense, NIMI is not a monitoring system but rather a measurement system.

- **Security**: NIMI has a strong authentication mechanism to protect NIMI platforms from unauthorized use. In addition, limits on the workload required to carry out tests can be controlled via the NIMI platform interface.

- **Extensibility**: One of NIMI's biggest strengths is its extensibility. The way NIMI is designed, it is easy to deploy new measurement modules. Therefore, it is conceptually straightforward to develop a new module, deploy it, and run new measurements.

- **Device Flexibility**: NIMI measurement is currently limited to end-host systems.

- **Multicast Independence**: NIMI does not rely on multicast for updating platforms/modules or for collecting measurement data.

- **Abstraction and Presentation**: NIMI does not have provisions for processing collected data; this function is outside the scope of the NIMI architecture.

GDT. The Globally Distributed Troubleshooting (GDT) system provides a mechanism to detect and report network problems across administrative domains[24, 25]. In GDT, each domain has a number of *expert modules*. Each module has associated areas of expertise. Expert modules in peer domains can contact each other and exchange problem reports with the goal of alerting remote domains of problems that may or may not be located in the remote network. Any entity within one domain may report a problem to an expert module. Expert modules then apply known domain-specific tests to confirm or deny the existence of the problem. GDT does not specify how to test or repair problems, it depends on locally available management systems for these operations. After confirming a problem, an expert module generates new *hypotheses* about potential causes of the problem and sends problem reports to other expert modules. If a problem is believed to exist in another domain, a hypothesis will be sent to an expert module in that domain. In this way, experts in peer administrative domains work together to locate actual problem points. GDT is designed as an application-layer, inter-domain debugging coordination tool. GDT's characteristics are as follows:

- **Intra- vs. Inter-Domain Support**: GDT is primarily designed to be an inter-domain tool. It is expected to facilitate communication and cooperation across administrative boundaries—a boundary across which management information typically does not flow.
- **Scalability**: GDT will likely not have to deal with many issues related to scalability. The situation in which a scalability bottleneck is created is when there are numerous error conditions that require expert modules to conduct and report on many debugging activities. If the infrastructure is at the point where there are numerous problems, worrying about the scalability of GDT is the least of a NOC's problems.
- **Security**: GDT does not specifically deal with the issue of security, but because of its focus on debugging, there is only minimal need for protection. The key weakness is the ability of an expert module to be attacked by receiving bogus requests for debugging information, or overwhelming it with debugging reports.
- **Extensibility**: GDT's need for extensibility revolves around the need to be able to handle new error conditions and to be able to use new debugging tools. To this extent, GDT is extensible but the complexity of effectively using new tools is likely to be considerable.
- **Device Flexibility**: As described, GDT expert modules can potentially run in any network device. As with MRM, capability of the expert module is dependent on the capability of the device on which it runs.
- **Multicast Independence**: GDT's communication mechanism does not use multicast and so management efforts are not hindered when multicast fails.
- **Abstraction and Presentation**: In GDT, the main data that has to be presented is the results of attempts to isolate problems and report time-to-resolution information.

6 Conclusions

In this paper we have attempted to understand the body of work related to multicast monitoring tools and systems. Beyond categorizing and describing many of the important tools that have evolved, we have focused on generalizing the paradigm used by many of these tools. Because a lack of adequate multicast monitoring tools has been cited as a reason for a lack of multicast deployment, we believe it is important to understand what the existing set of tools does and does not do. To this end, we have presented a generic architecture which represents the basic organization of many of these tools. We have also identified a set of metrics that can be used to further characterize existing monitoring efforts. Applying this set of metrics to a tool can help formalize the tool's advantages and disadvantages. As a case study, we use our work in reachability monitoring to show the kinds of results that are being produced; where current systems are weak; and how incremental work has tried to solve these problems. Our general conclusion is that adding support to the network for any kinds of service beyond best-effort delivery of IP packets is fundamentally hard. And while there are numerous debugging, management, and modeling tools available, each is only marginally effective in making multicast easier to deploy.

References

[1] S. McCreary and K. Claffy, "Trends in wide area IP traffic patterns, a view from Ames Internet exchange." http://www.caida.org/outreach/papers/AIX0005.

[2] K. Almeroth, "The evolution of multicast: From the MBone to inter-domain multicast to Internet2 deployment," *IEEE Network*, vol. 14, pp. 10–20, January/February 2000.

[3] B. Fenner and et al., *mrouted 3.9-beta, mrinfo, and other tools*, March 1998. Available from ftp://ftp.parc.xerox.com/pub/net-research/ipmulti/.

[4] W. Fenner and S. Casner, "A 'traceroute' facility for IP multicast." Internet Engineering Task Force (IETF), draft-ietf-idmr-traceroute-ipm-*.txt, August 1998.

[5] A. Swan, D. Bacher, and L. Rowe, *rtpmon 1.0a7*. University of California at Berkeley, January 1997. Available from ftp://mm-ftp.cs.berkeley.edu/pub/rtpmon/.

[6] H. Schulzrinne, S. Casner, R. Frederick, and J. V., "RTP: A transport protocol for real-time applications." Internet Engineering Task Force (IETF), RFC 1889, January 1996.

[7] D. Makofske and K. Almeroth, "Real-time multicast tree visualization and monitoring," *Software–Practice & Experience*, vol. 30, pp. 1047–1065, July 2000.

[8] D. Makofske and K. Almeroth, "MHealth: A real-time graphical multicast monitoring tool for the MBone," in *Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, (Basking Ridge, New Jersey, USA), June 1999.

[9] K. Auerbach, *DWTNDA: Dr. Watson, The Network Detective's Assistant*, November 1997. Available from http://www.cavebear.com/dwtnda/spd.html.

[10] S. Deering and D. Cheriton, "Multicast routing in datagram internetworks and extended LANs," *ACM Transactions on Computer Systems*, pp. 85–111, May 1990.

[11] J. Robinson and J. Stewart, *MultiMON 2.0 – Multicast Network Monitor*, August 1998. Available from http://www.merci.crc.ca/mbone/MultiMON/.

[12] D. Massey and B. Fenner, "Fault detection in routing protocols," in *International Conference on Network Protocols (ICNP)*, (Toronto, CANADA), November 1999.

[13] K. Sarac and K. Almeroth, "Monitoring reachability in the global multicast infrastructure," in *International Conference on Network Protocols (ICNP)*, (Osaka, JAPAN), November 2000.

[14] K. Sarac and K. Almeroth, "Supporting the need for inter-domain multicast reachability," in *Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, (Chapel Hill, North Carolina, USA), June 2000.

[15] P. Rajvaidya and K. Almeroth, "A scalable architecture for monitoring and visualizing multicast statistics," in *IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM)*, (Austin, Texas, USA), June 2000.

[16] T. Munzner, E. Hoffman, K. Claffy, and B. Fenner, "Visualizing the global topology of the MBone.," in *IEEE Symposium on Information Visualization*, (San Francisco, California, USA), October 1996.

[17] B. Huffaker, E. Nemeth, and K. Claffy, "Otter: A general-purpose network visualization tool.," in *INET*, (San Jose, California, USA), June 1999.

[18] K. Almeroth, L. Wei, and D. Farinacci, "Multicast reachability monitor (MRM)." Internet Engineering Task Force (IETF), draft-ietf-mboned-mrm-*.txt, October 1999.

[19] K. Almeroth and L. Wei, "Justification for and use of the multicast routing monitor (MRM) protocol." Internet Engineering Task Force (IETF), draft-ietf-mboned-mrm-use-*.txt, March 1999.

[20] J. Walz and B. Levine, "A hierarchical multicast monitoring scheme," in *International Workshop on Networked Group Communication (NGC)*, (Palo Alto, California, USA), November 2000.

[21] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser, "Protocol operations for version 2 of the simple network management protocol (SNMPv2)." Internet Engineering Task Force (IETF), RFC 1905, January 1996.

[22] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser, "Structure of management information for version 2 of the simple network management protocol (SNMPv2)." Internet Engineering Task Force (IETF), RFC 1902, January 1996.

[23] A. Rubens, C. Ravishankar, D. Thaler, A. Adams, B. Norton, and J. DiGiuseppe, "Merit SNMP-based MBone management project." http://www.merit.edu/~mbone/.

[24] D. Thaler and C. Ravishankar, "An architecture for inter-domain troubleshooting," in *Proceedings of Sixth International Conference on Computer Communications and Networks*, (Las Vegas, Nevada, USA), September 1997.

[25] D. Thaler, "Globally distributed troubleshooting (GDT): Protocol specification." Internet Engineering Task Force (IETF), draft-thaler-gdt-*.txt, January 1997.

[26] R. Malpani and E. Perry, *mmon: A multicast management tool using HP OpenView*, December 1999. Available from http://www.hpl.hp.com/mmon/.

[27] J. Case, R. Mundy, D. Partain, and B. Stewart, "Introduction to version 3 of the internet-standard network management framework." Internet Engineering Task Force (IETF), RFC 2570, April 1999.

[28] D. Agarwal and S. Floyd, "Tool for debugging internet multicast routing," in *Computer Science Conference*, (Phoenix, Arizona, USA), pp. 22–29, March 1994.

[29] A. Ghosh and P. Brooks, *MWATCH 3.6.2.* University College London, June 1994.

[30] B. Mah, "Measurements and observations of IP multicast traffic," Tech. Rep. UCB/CSD-93-735, University of California at Berkeley, March 1993.

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

[32] K. Almeroth, "A long-term analysis of growth and usage patterns in the Multicast Backbone (MBone)," in *IEEE Infocom*, (Tel Aviv, ISRAEL), March 2000.

[33] M. Yajnik, J. Kurose, and D. Towsley, "Packet loss correlation in the MBone multicast network," in *IEEE Global Internet Conference*, (London, ENGLAND), November 1996.

[34] M. Handley, "An examination of MBone performance," Tech. Rep. ISI/RR-97-450, Information Sciences Institute (ISI), University of Southern California (USC), January 1997.

[35] A. Adams, R. Bu, R. Caceres, N. Duffield, T. Friedman, J. Horowitz, F. Lo Presti, S. Moon, V. Paxson, and D. Towsley, "The use of end-to-end multicast measurements for characterizing internal network behavior," *IEEE Communications*, May 2000.

[36] V. Paxson, J. Mahdavi, A. Adams, and M. Mathis, "An architecture for large-scale internet measurement," *IEEE Communications*, August 1998.

[37] R. Chalmers and K. Almeroth, "Modeling the branching characteristics and efficiency gains of global multicast trees," in *IEEE Infocom*, (Anchorage, Alaska, USA), April 2001.

[38] M. Handley, "SAP: Session announcement protocol." Internet Engineering Task Force (IETF), draft-ietf-mmusic-sap-*.txt, March 2000.

[39] M. Handley, *SDR: Session Directory Tool.* University College London, November 1995. Available from ftp://cs.ucl.ac.uk/mice/sdr/.

[40] T. Friedman, R. Caceres, K. Almeroth, and K. Sarac, "RTCP reporting extensions." Internet Engineering Task Force (IETF), draft-ietf-avt-rctp-report-extns-*.txt, March 2000.

[41] K. Obraczka, "Multicast transport mechanisms: A survey and taxonomy," *IEEE Communications*, vol. 36, January 1998.

[42] E. Al-Shaer and Y. Tang, "Toward integrating IP multicasting in internet network management protocols," *Computer Communications–Integrating Multicast into the Internet*, 2000.

[43] K. Stephen and R. Atkinson, "IP authentication header." Internet Engineering Task Force (IETF), draft-ietf-ipsec-auth-header-*.txt, July 1998.

[44] R. Rivest, "The MD5 message-digest algorithm." Internet Engineering Task Force (IETF), RFC 1321, April 1992.