Remote Multicast Monitoring Using the RTP MIB

Julian Chesterfield, Bill Fenner, and Lee Breslau

AT&T Labs—Research
75 Willow Road
Menlo Park, CA 94025
{julian,fenner,breslau}@research.att.com

Abstract. Multimedia applications often involve one-to-many, or many-to-many, communication. These applications can be supported efficiently with network layer multicast. While multicast is a promising technology, monitoring multicast routing infrastructure and the performance of applications that use it presents challenges not found with unicast. In this paper we describe a monitoring architecture, and an implementation of the architecture, that uses application level monitoring to assess the performance of multicast-capable infrastructure. The architecture makes use of standards-based management technologies and allows both active and passive monitoring. The implementation has been tested extensively and provides a network operator with a network-wide view of multicast performance.

1 Introduction

IP multicast is an efficient way to provide support for one-to-many communication. Multicast was first used for distribution of real-time multimedia content in the Internet nearly a decade ago [6]. It experienced rapid deployment in the early 1990s, and it has been an active area for protocol development, resulting in several intra- and inter-domain protocols (e.g, DVMRP [9], PIM [8], MSDP [18]). Despite its initial promise and its ability to enable new applications, its growth slowed and it has yet to see widespread use in the Internet. Several issues have inhibited the deployment of multicast, including pricing models, security, and the complexity of routing protocols. One factor in particular, that has prevented wider deployment, is a lack of adequate management tools available to network operators. Without the kinds of tools available in the unicast domain, network operators find it difficult to monitor and debug their multicast networks.[1]

Management and monitoring of multicast networks is an inherently hard task. In contrast to the unicast domain, where a mature set of tools exists, multicast

The original version of this chapter was revised: The copyright line was incorrect.This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-540-45812-8_28

[1] For simplicity, we use the term *multicast network* to refer to an IP network in which multicast forwarding is enabled on the routers. For the purposes of this paper, the particular multicast routing protocol employed – DVMRP, PIM, MOSPF, etc. – is not important.

K.C. Almeroth and M. Hasan (Eds.): MMNS 2002, LNCS 2496, pp. 154–169, 2002.

presents some unique challenges. First, whereas unicast traffic generally follows a single end-to-end path between a source and destination, multicast distribution involves transmission along a tree from a single sender to multiple receivers. Further, membership in a multicast group can be dynamic, so the tree itself can change over the lifetime of a multicast session, even if the topology is static. Finally, in contrast to unicast, the state needed to forward a multicast packet is not instantiated until a session is initiated, making it difficult to determine the status of multicast routing until traffic is flowing.

Early attempts to provide management functionality for multicast networks resulted in a set of ad hoc tools. These included *mtrace* [11], which provides information about the set of links on the path from a sender to a single receiver in a multicast tree, and *mrinfo* [19], a utility that returns information about a multicast router's local configuration. These tools provided only limited information about multicast configuration and operation, and were not always implemented on all multicast routing platforms. More recently, tools such as *mhealth* [14], which discovers and displays an entire multicast tree, have provided additional information for network operators. Similarly, multicast-related MIBs, which have been defined (e.g., [12,15,16]), further enhance the set of tools available for managing multicast. Nonetheless, the state of network management tools for multicast lags far behind what is available for unicast infrastructure and continues to impede deployment.

In this paper we present an architecture for remote management of multicast routing infrastructure and describe our implementation of this architecture. The goal of our work was to provide a tool that would give a network operator, from a single management station, a system wide view of the end-to-end performance of a multicast network. In designing this system, we were guided by several key requirements. First, we wanted a system that would be based on standard protocols. This facilitates integration with other management tools and leverages existing technology. Second, we desired a clean separation of functionality between data collection and data analysis. The purpose of this separation is to enable a data collection architecture based on Internet standards, while enabling independent development of analysis and management tools. Third, we wanted a monitoring architecture that allowed both passive monitoring of existing multicast sessions, while also permitting creation of test sessions and the artificial generation of multicast traffic. Finally, we wanted a system that supported both long term background monitoring, as well as active debugging of multicast sessions.

Our architecture and implementation, which we call `RMPMon`, meet these requirements. By combining known standard technologies, and extending existing management standards to enable a novel method of generating test traffic, our system can provide an operator responsible for managing a multicast network with better information than was previously available. Because it is based around standard technology, we believe `RMPMon` can facilitate the development of additional management tools for multicast and help further deployment of this promising technology. The rest of this paper is organized as follows. In the

following section, we present relevant background about the standard protocols that we make use of in our architecture. In Sect. 3, we describe extensions to these protocols that provide additional building blocks for our system. The architecture, implementation and evaluation of our tool are presented in Sect. 4. We discuss related work in Sect. 5 and conclude with a discussion of future work in Sect. 6.

2 Background

We built `RMPMon` around two key Internet protocols: the Simple Network Management Protocol (SNMP) [5] and the Real-Time Transport Protocol (RTP) [27]. Using SNMP has several benefits. As the standard management protocol for IP networks, it is widely available in existing network management platforms, which provides the opportunity to integrate our tool into these platforms. In addition, we are able to leverage important features in SNMP, such as its security model, and we can take advantage of existing public domain implementations and focus on those pieces of functionality unique to our system. Finally, SNMP uses unicast transport, so we do not require multicast to be deployed everywhere between our management station and the systems we are monitoring.

RTP is a transport protocol for real-time applications which supports unicast and multicast applications. In the remainder of this section we present a brief overview of SNMP and RTP to provide background for the remainder of the paper.

2.1 SNMP

SNMP is used to manage a wide range of devices, protocols and applications in the Internet. Management information can pertain to hardware characteristics, configuration parameters or protocol statistics. For example, SNMP can query a router to determine the bandwidth of one of its interfaces or the number of packets that it has forwarded. SNMP is built around the concept of a Management Information Base, or MIB, which defines a set of managed objects for a device or protocol. A MIB consists of a set of objects, or MIB variables, related to the device or protocol being managed. Each object has a unique name, syntax and encoding, as well as a set of properties. The properties associated with an object determine whether the object is readable, writable, or both. Managed objects may be either static or dynamic. Static objects always exist while dynamic objects are created or deleted as a protocol or device changes state. For example, in the TCP MIB, the number of active TCP connections is a static object, whereas an object representing the state of a particular connection is dynamic.

A network management station uses SNMP to manage a remote device by communicating with an agent running on that device. The remote agent interacts directly with the managed objects on that device. SNMP defines several commands that are used between the management station and the remote agent. There are two categories of SNMP commands:

– The value of MIB variables are retrieved from a remote agent using the commands Get, GetNext, and GetBulk.
– The value of MIB variables can be set using the Set command.

In the simplest cases, the Set command can be used to configure parameters of a protocol or device. However, setting MIB variables can also cause other actions to be taken, making it a powerful tool in managing a remote client. This functionality is a key enabler in the system we describe later. We next describe how SNMP is used in the context of the RTP protocol.

2.2 RTP and Related MIBS

RTP is a transport protocol intended for use by real-time applications, such as video-conferencing and playback of recorded multimedia content [27]. Because these applications often involve more than two participants, RTP was defined to accommodate multicast communication. An RTP session is identified by a destination address, either multicast or unicast, and a destination port number. Within the context of multicast, a host can join the relevant multicast group to become a receiver of a session, or a host can transmit packets to the session address to become a sender to a session, or both.

Real-time sessions require timely delivery of data for the data to be useful, and these applications are assumed to be tolerant of modest amounts of packet losses. In addition, feedback in the form of acknowledgments or requests for retransmission presents scalability problems for large multicast sessions. For these reasons, RTP does not provide reliable data transport. Nonetheless, some feedback among session participants can be useful to RTP-based applications, as a means to share state about performance, to identify sources, and for monitoring purposes. This functionality is provided by a companion protocol, RTCP. Session participants (both senders and receivers) periodically transmit RTCP packets. These packets are sent to the destination IP address of the session using a different port number than the data packets.

RTCP control packets include the following information:

– Each participant periodically transmits identifying information, such as user name and email address.
– Each receiver periodically transmits condition reports. These reports include the loss rate and an estimate of delay variance that the receiver experiences for each source.

Thus, every session participant learns the identities of all active participants as well as the loss rate and other performance statistics experienced by each receiver.[2]

[2] In order to limit bandwidth consumed by the control protocol, the frequency at which a participant transmits RTCP packets is inversely proportional to the number of session participants. The utility of RTCP reporting with very large sessions is an open question, and several ideas to address the scalability question have been proposed.

Management information for RTP is defined in an associated MIB [4]. The RTP MIB is organized into 3 tables.[3] These are the session table, the sender table, and the receiver table. Each entry[4] in the session table contains information about an active session at the host. Each entry in the sender table contains information about a sender to one of these active sessions. The session and sender fields provide a unique index for each entry in the sender table. Finally, each entry in the receiver table contains information about a single sender/receiver pair for each session. The session, sender and receiver fields provide a unique index for entries in the receiver table. The objects represented in these tables are all dynamic. That is, they are instantiated when a host joins a session, when a new sender is identified, or when a receiver report from a new session is received, respectively. The MIB variables in the receiver table include such things as the number of packets received, the number lost, and the measured delay jitter.

SNMP is used to query or set the values of these MIB variables. For example, by querying the receiver table, a management client can determine the loss rate experienced between a particular sender/receiver pair. In addition, using the Set command, a management client can create new entries in these tables. By creating an entry in the session table and setting a MIB variable to make that entry active, a management client can cause a remote agent to join a multicast group as a receiver. We use this functionality extensively in the management framework that we describe in Section 4.

3 RTP Sender MIB

RTP and the RTP MIB provide necessary building blocks for our management system. However, they are not sufficient to enable the entire range of functionality we required. Therefore, we defined a new MIB that extends the capability of multicast management to enable active monitoring. We refer to this new MIB as the RTP Sender MIB.

The RTP Sender MIB is an extension to the RTP MIB described above. The MIB describes data streams that are generated by a remote agent. Specifically, it contains a table describing the characteristics of the data streams, and links the source to an existing entry in the RTP MIB. Entries are indexed by a test name and a user name to provide user based access control. Each entry contains the following objects: packet generation rate, packet length, and outgoing interface. When an entry is set to active, the remote agent begins transmitting packets according to the parameters in the sender entry.

Allowing the generation of test traffic by remote agents has certain inherent security risks. This problem is addressed with a view based access control mechanism that is implemented, as described above, in the MIB. Only authenticated

[3] RFC 2959 actually defines 6 tables. The latter three provide reverse lookups to enable efficient indexing. Since they provide performance improvements, and do not directly address issues of functionality, we do not discuss them further here.

[4] In SNMP terminology, a table entry is referred to as a row.

users are able to create active streams on the agent, where the authentication is part of the SNMP security framework.

The active sender MIB provides an important function to the monitoring architecture. The management client can create a new entry in a remote agent's sender table and populate the MIB variables with appropriate packet generation statistics. The remote agent then generates traffic according to these parameters when the management client sets the entry to become active. In this way, the management client can generate test traffic at one or more remote agents and monitor receipt of this traffic using the RTP MIB.

4 RTP-Based Monitoring System

We now describe the management and monitoring system that we have developed. We begin with a high level description of the architecture of the system and its features. We next describe our implementation. Finally, we describe our experience with the implementation and present results of measurements we've gathered in our lab.

4.1 Architecture

The system we developed consists of two key components: remote agents running on managed devices (hosts or routers) that participate in the monitoring system and a management client. The remote agents implement the RTP MIB and the RTP Sender MIB, and they communicate with the management station using SNMP. In response to SNMP commands, they join RTP sessions as senders or receivers. As senders, they generate test traffic for controlled experiments. As receivers, they receive RTP data packets and RTCP control packets, and they maintain statistics about session participants and performance. Since these data are part of the RTP MIB, they can be returned to the management client in response to SNMP queries.

The management client communicates with remote agents via SNMP. The management client controls the monitoring by causing remote agents to join sessions as senders or receivers, and the remote agent collects statistics from the remote agents. The management station is then able to analyze the data and present it to a network operator in useful ways.

This architecture has several desirable features. First, the separation of functionality between the remote agents is a flexible design that enables extensibility. Rather than designing a monolithic system in which the data collection and analysis are tightly coupled, we have separated the two primary components. Hence, new management applications can be written that use the basic functionality provided by the remote agents in new and interesting ways.

Another key feature of the architecture is that by incorporating the RTP Sender MIB, it is capable of both active and passive monitoring. Passive monitoring is used to monitor actual user multicast sessions. In this case, the management client causes one or more remote agents to join the session as receivers.

These receivers monitor session performance (i.e., loss rate, delay jitter), and send and receive RTCP messages to and from other session participants.[5] Reporting information can then be returned to the management station. Note that RTCP receiver reports are transmitted to the session's multicast address and are seen by all receivers in a session. Therefore, by creating a single receiver at a remote agent, the monitoring system is able to collect information about all senders and receivers in a session, regardless of their location in the network. This presents the network operator with a network-wide view at very little overhead beyond the existing session traffic. Note, however, that even though RTCP receiver reports are distributed to all group members, using more than one remote agent to collect and report data may be useful. As an example, querying MIB values of more than one session participant allows consistency checking, which can identify situations in which routing problems prevent distribution of receiver reports to all session participants.

Passive monitoring provides performance information for actual user sessions. However, one of the difficulties of monitoring multicast infrastructure is that it is hard to assess performance when sessions are not active. Active monitoring fills this void. With active monitoring, rather than joining a user multicast session, a special session dedicated to monitoring is created. The management client uses the RTP MIB and RTP Sender MIB to create receivers and senders for this testing session. The traffic generation parameters of the sender or senders are controlled by the management station. In this way, the network operator is given a view of performance that is independent of user sessions.

This system can be used for both long term background monitoring as well as for reactive debugging. In the former case, a long running test session could be used to collect performance statistics over time. In the latter, network operators who are alerted to potential problems can create active sessions to help debug network problems.

4.2 Implementation

We have implemented the architecture described above in order to test our ideas and gain experience with them in both laboratory and production environments. One of our design goals was to leverage prior work and experience, concentrating our effort on those parts of the design unique to our system and easing the task of porting our implementation to other platforms. As such, we have taken extensive advantage of open source implementations that provide some of the needed functionality. In this section, we describe the implementation of both the remote agents and management station.

The remote agent is built within the AgentX framework [7], which supports extensible SNMP agents. Specifically, it allows a single master SNMP agent to run on a client and dispatch SNMP commands to subagents. AgentX defines the

[5] We anticipate that future extensions to the RTP MIB will allow the network operator to make a remote agent's participation in the session invisible to other session participants by setting its TTL to zero.

interface between the master and subagents. We implemented the remote agent as an AgentX subagent. The remote agent implements both the RTP MIB and RTP Sender MIB, and registers these MIBs with the master SNMP agent on the host. It also implements an RTP application. For the SNMP functionality, we used Net-SNMP, an open source SNMP library [20]. For the RTP functionality, we used the UCL RTP library, which implements RTP (including RTCP) for multicast as well as unicast sessions [23]. The remote agent is capable of acting as a sender or receiver of multiple RTP sessions. The application extracts relevant information to populate the MIB with session conditions in real-time.

We also implemented an application to serve as our management client. This application was built by extending the *Scotty* management application [28]. *Scotty* includes extensions to TCL that provide access to SNMP functions. In addition, it incorporates *tkined*, a graphical network editor designed for management applications. Thus, a user can manipulate a graphical user interface depicting agents in the network that are being managed. User interface actions then cause SNMP commands to be issued by the management client to the remote agents. Data collected by the management client can then be displayed in the user interface. We added functionality to *tkined* to enable management of the RTP agents. Scripts were written to allow simple activation of pre-configured sessions for debugging and real-time monitoring purposes. Additional functionality was added for monitoring existing RTP sessions.

4.3 Experience

`RMPMon` was originally developed on FreeBSD and has been ported to Solaris and HP-UX. We have tested it extensively in our lab, and it is now being evaluated for use by a major ISP. In this section, we describe its use in more detail and show results from its use.

Experiments. As part of our debugging and analysis of the tool, we have performed many experiments with it in our lab. We now describe one such test. We use this test to demonstrate some of the tool's functionality and to gain some understanding of the overhead associated with it. We will then describe in less detail other uses of the tool.

Figure 1 shows a snapshot of the management client's graphical interface. The network operator can use this interface to dynamically create an object based network map and associate these objects with network devices. The map in the figure shows the topology used in the experiment we describe here. The test network consists of 14 Pentium III class PCs running FreeBSD 4.3. Three of these (cubix09, cubix10, and cubix11) are configured to act as routers for both unicast and multicast traffic. The remaining 11 are end hosts attached to one of six subnets. All host interfaces provide 100 Mbit/s switched ethernet connectivity. In order to simulate conditions of network loss, we used the *dummynet* software [10] to create artificial loss. *Dummynet* implements a configurable traffic shaper based on the FreeBSD firewall code, enabling a user to create virtual traffic

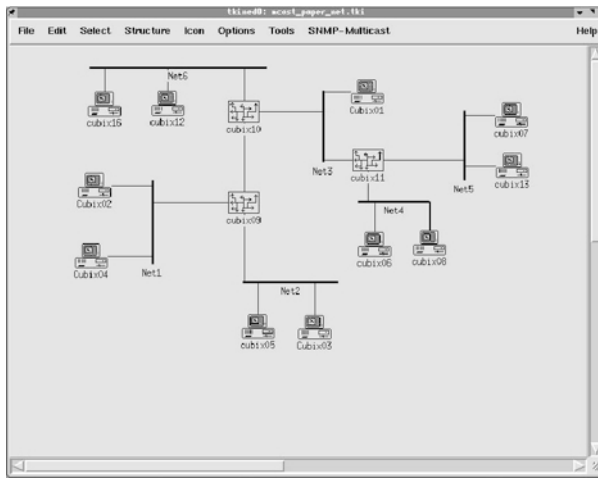

Fig. 1. Management Configuration

Fig. 2. SNMP-Multicast Menu

flow *pipes* based on a source and destination identifier and to configure certain characteristics for those flows. In this case, we used the pseudo-random fixed packet loss rate parameter to introduce 4% loss on the outgoing link from cubix09 to cubix10, the outgoing link from cubix10 to Net3, and the outgoing link from cubix11 to Net5. In this way, traffic between any pair of end hosts would traverse 0, 1, 2 or 3 lossy links.[6]

Once the network map is configured (Fig. 1), the experiment is controlled through the SNMP-Multicast pulldown menu shown in Fig. 2. For example, to start an RTP listener on a remote agent, the operator selects an object (e.g., cubix02) and then selects Start RTP Listeners on the pulldown menu. Similarly, remote senders are started via the Start RTP Senders option. For this experiment, we configured cubix02, cubix05, cubix07, and cubix16 as senders, each generating 64 kbps data streams. All of the end hosts were configured as receivers, and they all participated in RTCP exchanges. Cubix04 was selected as the host to be queried by the management client.[7]

The output of the experiment is shown in Figs. 3 and 4. Figure 3 shows a *Reception Quality Matrix*. We developed this tool by integrating the interface from the Reception Quality Matrix (which was originally implemented in version 4 of the Robust Audio Tool [22]) into our management client. The tool reports on connectivity between all senders and receivers, with each column representing a sender and each row representing a receiver. Each cell in the matrix indicates the loss rate between a sender and receiver. In addition, color codings indicate

[6] Configuring the loss is not done with the management tool. Rather, this is part of the general network setup for the experiment.

[7] Recall from Section 2.2 that we only need to query a single remote agent since all remote agents exchange RTCP receiver reports.

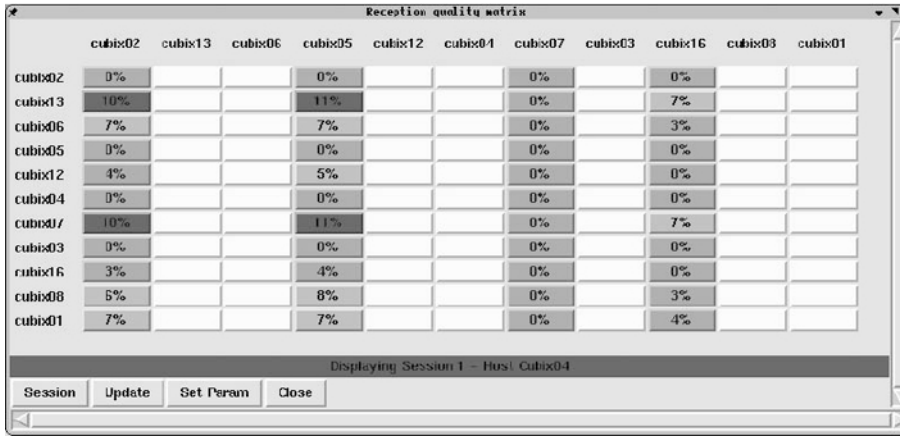

Fig. 3. Reception Quality Matrix

0-5% loss (green), 5-10% loss (orange), and loss over 10% (red), to assist in easy identification of problems.

Looking at Fig. 3, we see that the results are consistent with the experimental setup. cubix07 and cubix13, the two hosts on Net5, experience high loss (10-11%) for packets transmitted by senders cubix02 and cubix05. The paths between these senders and receivers traverse all three of the lossy links.[8] Paths that traverse two of the lossy links result in more moderate loss (e.g., 7% loss between cubix02 and cubix01), and single loss paths yield 3%-5% loss (e.g., cubix16 to cubix01). In all cases, hosts that are connected by paths that do not include links with loss report 0% loss.

In addition to reporting performance conditions in different places of the network, the tool can be used to deduce where in the network loss is occurring. For example, given that there is only a single link between cubix16 and cubix01, one can conclude that it is this link that is responsible for the loss. The loss between cubix02 and cubix01 can then be attributed in part to loss on the aforementioned link, and additional loss experienced on the link between cubix09 and cubix10.

A second tool that we developed is the *Loss Graph* tool which provides another view of loss rates. It periodically queries an agent to build up a series of values over time to indicate longer term performance of the agents. The tool takes a single source/receiver pair, and plots the reported loss at a user specified query interval. It also provides the facility to display the results over a variety of time scales. Figure 4 indicates the loss reported by host cubix07 from source cubix02 over a half hour time scale. The long timescale helps to draw attention to the trends over time, something that is not apparent from the snapshot results provided by the *Reception Quality Matrix*.

[8] Loss on these links is random, accounting in part for the slight variation between the expected loss rate, which is approximately 12%, and the reported loss rates.

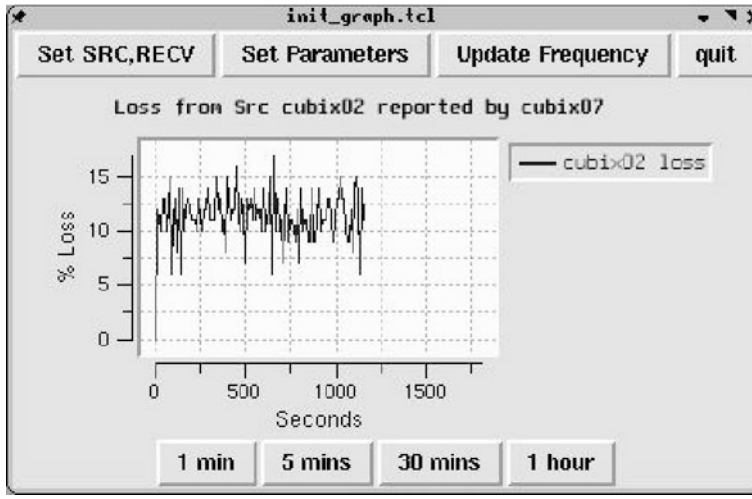

Fig. 4. Loss rate

Overhead. The kind of experiment described above could be used by a network operator to monitor performance in a production network. It is important to consider the overhead created by such monitoring. There are two kinds of overhead: packets generated by the test traffic (RTP and RTCP), and packets associated with SNMP traffic between the management client and the remote agents. The amount of RTP traffic is directly controlled by the network operator, who determines both the number of senders to instantiate and the rate at which they send. On network links of even moderate speed, it should be possible to accurately measure loss using fairly modest data rates.

In order to gain some understanding of the overhead caused by the SNMP traffic we measured the SNMP traffic between the management client and remote agents. The average aggregate rate for this traffic (including both directions) was approximately 4 kbps. However, this data is very bursty as a result of the management station periodically polling the remote agents. Polling resulted in transfer of about 10 kbytes of data, followed by a quiescent period. The overhead of SNMP traffic will obviously be impacted by the frequency of the polling and by the number of participants in the session being monitored.[9] While we can't draw

[9] The overhead need not grow linearly with the number of session participants. The tool queries a single session participant, and there is some fixed overhead associated with this query. The incremental cost of returning information about another receiver should be small. Further, for very large groups, the management client can have the flexibility to request information about a subset of session participants. Also note that for active monitoring, `RMPMon` need not scale to very large number of remote agents. A network operator can monitor performance across its network by deploying one remote agent per routing center, likely on the order of several tens of locations.

firm conclusions from one experiment, the results indicate that the overhead of the tool is likely minimal.

Further, we have not optimized the performance of our implementation. SNMP provides flexibility in how data is retrieved from remote agents. For example, an entire table can be retrieved with the GetBulk command, or a single object can be retrieved in one message. The optimal strategy depends on several factors, such as the percentage of objects in a table which are needed by the management station. We have not paid particular attention to these issues, concentrating on functionality rather than efficiency in our implementation. Simple analysis of packet traces in our experiments indicates that we can easily achieve a factor of 2 reduction in the overhead with a more careful implementation.

The decision to retrieve data from a single host connected to the session was motivated by the network configuration as well as implementation simplicity as far as SNMP management is concerned. This would not help in situations where RTCP packet loss is very high, or indeed where multicast connectivity is broken. For a small increase in SNMP traffic overhead, an efficient management client could selectively ask each member of the session for its own specific values. In the case of calculating loss, the RTP agent would return the MIB variables denoting lost packets and expected packets for each source that it hears. In this way, GetBulk requests might still be used for sessions with multiple senders to maintain low SNMP packet overhead, whilst retrieving more accurate information for the management station. This would provide the additional advantage of helping to detect active sources and receivers that are not ubiquitously heard by all members. A simple table view would highlight these inconsistencies.

Other Usage Scenarios. Above we described an experiment that demonstrated how `RMPMon` could be used to measure application level performance across a multicast network. We now mention some other ways in which it can be useful.

`RMPMon` can be used for passive monitoring of an actual user multicast session. In this case, the management station need only create one receiver somewhere in the network. By participating in the RTCP exchanges, this receiver collects session-wide information, which is returned to the management station via SNMP. This scenario demonstrates one of the advantages of our approach. If the receiver is located along the existing distribution tree, then no additional RTP traffic is distributed and no additional forwarding state is created. Using SNMP to retrieve reporting statistics from the agent is more efficient that having the management station receive both the RTP data packets and the RTCP control packets.

A second way in which `RMPMon` can be used is for consistency checking. In this scenario, every remote agent is asked for a list of active sources that it hears in the session. A simple consistency check across the data returned can indicate whether there is full, partial or no multicast connectivity to all relevant areas of the network.

RMPMon could be used in conjunction with other debugging tools. For example, *mtrace* can be used to debug problem areas in the case of poor connectivity between clients. This action can be initiated from the Reception Quality Matrix display. It might also help to identify routing anomalies in case data does not flow over the intended or preferred path.

RMPMon can also be used in conjunction with other tools that monitor multicast routing performance. For example, MSDP Source Active messages are triggered by data packets. If MSDP messages are being monitored, RMPMon can be used to generate multicast traffic at remote places in the network. A network manager can then verify that the proper MSDP messages are seen in MSDP protocol exchanges. Instantiating remote senders and receivers can also be used to verify that distribution trees are correct. For example, a receiver could be created and then terminated in order to verify that pruning is working correctly in a protocol such as DVMRP.

Additional management applications could be developed to extend the use of the tool. For example, rather than continuously displaying the reception quality matrix, the management client could monitor when certain thresholds of performance (e.g., loss rate) are crossed and alert the network operator.

5 Related Work

There has been a significant body of work in the area of multicast management and monitoring that has resulted in tools to support network, application, and session layer monitoring. Here we briefly review the work most closely related to ours. The interested reader is referred to [26] for a more in depth review.

The need for management tools became evident with the rapid growth of the MBone in the early 1990s. *Mrinfo* was an early tool that queried multicast routers for their adjacencies. Using this tool, one could then build a map of the multicast topology. Multicast traceroute, or *mtrace*, defined a protocol for querying routers along a path from receiver to source. This could be used to determine the path from sender to receiver, and ultimately an entire distribution tree. In addition, it can provide statistics about the number of multicast packets, and their loss rate, traversing a link. *Mtrace* remains an effective tool for debugging multicast reachability problems. Mantra [21] is a more recent system that determines multicast routing information by querying routers for this information. These tools all have the characteristic that they operate at the network level and retrieve adjacency or routing information from the routers themselves. Our work takes the alternative, and complementary approach, of monitoring performance information at the application level. There are tradeoffs associated with both approaches [21] and we believe that both kinds of tools are needed.

The earliest multicast debugging tools were developed outside the context of a general management framework. More recently, MIBs have been defined for multicast protocols. Tools such as *mrtree*, *mstat* and *mview* [17] are based on SNMP and take advantage of information in these multicast-related MIBs. As such, they represent a logical progression in the development of multicast tools.

RMPMon is also based on SNMP, and provides functionality not present in other SNMP-based tools.

The use of multicast in SNMP for communication between management stations and remote agents has been proposed [1]. This work is largely complementary to ours, as we have not focused on the communication channel between the management station and remote agents. One similarity is that both have mechanisms whereby the management station can cause a remote agent to join multicast groups. In RMPMon the multicast groups are used for communication between remote agents while in this other work the multicast groups are used for communication between the management station and the remote agents.

Ours is by no means the first work to use application level monitoring generally, or RTP-based monitoring, specifically. Rtpmon, RQM and *mhealth* are 3 tools that use RTCP reporting information in various ways. Rtpmon [24], developed at UCB, collects RTCP reporting information on a host and presents a loss matrix similar to the one we showed in Section 4.3. RQM, which provided the interface we used for our loss monitoring matrix, also collects RTCP reporting information for a session. *Mhealth* [14] is a tool that integrates both network layer information and application layer information to present both the multicast topology and end-to-end performance information. Our work extends the idea of using RTCP receiver reports embodied in all of these tools and embeds it into an SNMP framework. Incorporating it into a standard management framework provides several advantages, not the least of which is the ability to instantiate remote agents to participate in the monitoring.

The Multicast Reachability Monitor (MRM) [3] has been implemented on some commercial router platforms, and it has been used in an integrated framework for multicast monitoring that includes topology discovery and performance monitoring [25]. MRM is a protocol for generating test traffic on remote agents, providing functionality similar to what we achieve with the RTP Sender MIB. It represents an important development in the ability to monitor multicast routing infrastructure. We have borrowed this concept and used it within the context of SNMP. This allows us to take advantage of the authentication and access control available in SNMP, and it allows integration into a tool that allows both active and passive monitoring. SMRM [2] is another tool that is very similar to RMPMon. Like the MRM-based tool, SMRM allows a central management station to generate test traffic remotely and to collect statistics about performance. The primary difference between that work and ours is the use of RTP and its MIB in RMPMon. The use of RTP provides two primary advantages. First, it allows passive monitoring of multicast sessions that use RTP. Second, we leverage the reporting functionality of RTCP so that the management station can receive information about all receivers by querying a single remote agent.

6 Conclusions

In this paper we described a new tool for monitoring and managing multicast networks. The tool has several desirable features. It is based on standard technol-

ogy and can thus be integrated with common management platforms. It enables remote monitoring of multicast infrastructure, using both passive and active tests. In addition, it does not depend on multicast connectivity between the management station and remote monitoring agents. We have demonstrated its use through testing in our lab environment, and it provides useful functionality at acceptable levels of overhead.

Our current implementation is a prototype that has served as a proof of concept. We see several opportunities for future work to build on the architecture and implementation we have presented here:

- Integrating new developments in the area of Source Specific Multicast (SSM) extensions would be a useful extension to the architecture. Monitoring SSM groups and providing the capability to generate IGMPv3 source include and exclude messages and monitor the resulting state in routers in the network will be key elements in managing single source distribution architecture networks. This will require updates to relevant MIBs in addition to implementation work on the remote agent software.
- We intended to integrate the functionality provided by the RTP Sender MIB into a generic traffic generation MIB currently being considered in the IETF [13].
- One of the reasons for basing our implementation on SNMP was to allow for integration of `RMPMon` into standard management platforms. For the prototype, we built a standalone management client. In the future, we will explore integrating our work into a common platform, such as HP Open View.

Multicast is a promising technology whose potential value has not yet been realized. The increasing demand for the delivery of multimedia content in the Internet may provide a push for multicast. If this is to happen, the long-known challenges of monitoring and managing multicast infrastructure will need to be solved. Several years of work have shown that this is indeed a difficult problem. We believe standards-based tools that allow for end-to-end performance monitoring of multicast infrastructure, like the one presented here, must be a part of the solution.

References

1. Ehab Al-Shaer and Yongning Tang. Toward integrating IP multicasting in Internet network management protocols. *Journal of Computer and Communications Review*, December 2000. 167
2. Ehab Al-Shaer and Yongning Tang. SMRM: SNMP-based multicast reachability monitoring. In *IEEE/IFIP Network Operations and Management Symposium*, Florence, Italy, April 2002. 167
3. Kevin Almeroth, Kamil Sarac, and Liming Wei. Supporting multicast management using the Multicast Reachability Monitor (MRM) protocol. Technical report, UCSB, May 2000. 167
4. M. Baugher, B. Strahm, and I. Suconick. Real-Time Transport Protocol Management Information Base. RFC 2959, Internet Engineering Task Force, 2000. 158

5. J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. Protocol operations for version 2 of the simple network management protocol (SNMPv2). Request for Comments 1905, Internet Engineering Task Force, January 1996. 156
6. Stephen Casner and Stephen Deering. First IETF Internet audiocast. *ACM Computer Communication Review*, 22(3):92–97, July 1992. 154
7. M. Daniele, B. Wijnen, M. Ellison, and D. Francisco. Agent extensibility (AgentX) protocol version 1. RFC 2741, Internet Engineering Task Force, January 2000. 160
8. S. Deering, D. Estrin, D. Farinacci, Van Jacobson, C.-G. Liu, and L. Wei. An architecture for wide-area multicast routing. In *SIGCOMM Symposium on Communications Architectures and Protocols*, London, UK, September 1994. 154
9. Stephen Edward Deering. *Multicast routing in a datagram internetwork*. PhD thesis, Stanford University, Palo Alto, California, December 1991. 154
10. Dummynet. http://info.iet.unipi.it/~luigi/ip_dummynet/. 161
11. Bill Fenner and Steve Casner. A traceroute facility for IP multicast. Internet Draft, Internet Engineering Task Force, July 2000. 155
12. Bill Fenner and Dave Thaler. Multicast Source Discovery protocol MIB. Internet Draft, Internet Engineering Task Force, July 2001. 155
13. C. Kalbfleisch, R.G. Cole, and D. Romascanu. Definition of managed objects for synthetic sources for performance monitoring algorithms. Internet Draft, Internet Engineering Task Force, March 2002. 168
14. David Makofske and Kevin Almeroth. MHealth: A real-time multicast tree visualization and monitoring tool. In *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Basking Ridge, New Jersey, June 1999. 155, 167
15. K. McCloghrie, D. Farinacci, and D. Thaler. Internet Group Management Protocol MIB. Request for Comments 2933, Internet Engineering Task Force, October 2000. 155
16. K. McCloghrie, D. Farinacci, and D. Thaler. IPv4 Multicast Routing MIB. Request for Comments 2932, Internet Engineering Task Force, October 2000. 155
17. Merit SNMP-Based MBone Management Project. http://www.merit.edu/~mbone/. 166
18. David Meyer and Bill Fenner. Multicast source discovery protocol (MSDP). Internet Draft, Internet Engineering Task Force, November 2001. Work in progress. 154
19. mrinfo. ftp://ftp.parc.xerox.com/pub/net-research/ipmulti/. 155
20. Net-SNMP. http://net-snmp.sourceforge.net/. 161
21. Prashant Rajvaidya and Kevin Almeroth. A router-based technique for monitoring the next-generation of internet multicast protocols. In *International Conference on Parallel Processing*, Valencia, Spain, September 2001. 166, 166
22. Robust Audio Tool. http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/. 162
23. UCL RTP Library. http://www-mice.cs.ucl.ac.uk/multimedia/software/common/. 161
24. Rtpmon. ftp://mm-ftp.cs.berkeley.edu/pub/rtpmon/. 167
25. Hassen Sallay, Radu State, and Olivier Festor. A distributed management platform for integrated multicast monitoring. In *IEEE/IFIP Network Operations and Management Symposium*, Florence, Italy, April 2002. 167
26. Kamil Sara and Kevin C. Almeroth. Supporting multicast deployment efforts: a survey of tools for multicast monitoring. *Journal of High Speed Networks*, 9(3,4):191–211, 2000. 166
27. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: a transport protocol for real-time applications. RFC 1889, Internet Engineering Task Force, 1996. 156, 157
28. Scotty. http://wwwhome.cs.utwente.nl/~schoenw/scotty. 161