Supporting Multicast Management Using the Multicast Reachability Monitor (MRM) Protocol

Kevin C. Almeroth
Dept of Computer Science
University of California
Santa Barbara, CA 93106
almeroth@cs.ucsb.edu

Kamil Saraç
Dept of Computer Science
University of California
Santa Barbara, CA 93106
ksarac@cs.ucsb.edu

Liming Wei
Siara Systems, Inc.
300 Ferguson Drive
Mountain View, CA 94043
lwei@siara.com

May 2000

Abstract

Network management for multicast traffic has become a key to the continued deployment of multicast in the Internet. The lack of usable management tools is believed to be one of the barriers to any substantial use of multicast in production Internet services. Because multicast is a relatively new service, management tools either do not exist; do not provide the right functions; or are too hard to use. As a result, there is a real need to develop new multicast management solutions. In this paper we propose a new protocol that provides mechanisms to actively manage and monitor multicast reachability and path quality. Our proposed protocol, called the Multicast Reachability Monitor (MRM), can be used to send, receive, and collect statistics about a multicast stream. This information can then be presented using novel visualization techniques. The ultimate goal is to identify problems before they happen, as they are happening, and/or to assist in troubleshooting efforts. From this perspective, MRM is designed to be used as a building block for an integrated multicast network management tool set. In describing MRM, we start by defining some important characteristics of a multicast management system. We also describe the MRM protocol components. After discussing likely management scenarios, we finish by describing the state of the MRM protocol and compare it to other available network management platforms.

1 Introduction

Management of multicast traffic has become a key technical barrier to the further deployment of multicast in the Internet. The lack of usable management tools is being listed among the reasons why there has not been more rapid deployment and use of multicast[1]. For sake of completeness, other barriers that have been listed include the relative instability of multicast routing protocols, the lack of substantive experience among network engineers, and the "chicken-and-egg" problem of needing a true "killer application". However, as most of these are not technical problems but administrative and deployment issues, they are less interesting to researchers. Furthermore, in many cases time will solve these problems. There are now several companies who are developing a wide variety of Internet services on top of multicast communication. The focus of our work is to develop techniques to support the robust deployment of multicast through multicast-capable network management techniques and tools.

Because multicast is a relatively new service in the history of networking, many of the existing tools have been developed specifically to assist with debugging new protocols. This has impacted the existing techniques and tools in a number of ways. These include:

- Those who have developed tools have only focused on support for very specific protocol functions. Instead of focusing on tools that support traditional network management services like fault isolation and detection, tools help experts configure networks and verify protocol operation.
- When Network Operations Center (NOC) personnel attempt to use the few existing tools, they find them hard to use and insufficient for traditional network management. The reason is not because gathering statistics is a particularly difficult task, but more because the data that is collected is difficult to analyze.
- For multicast, like any new technology, there exists a lack of widespread understanding and expertise. NOC personnel tend to have a broad range of knowledge instead of expertise in a few areas. As a result, managing multicast traffic requires necessary mechanisms to provide support for management functions and it requires easy-to-understand and easy-to-use management tools implemented on top of these mechanisms.

From a management point-of-view, successfully deploying multicast requires the ability to have 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. Currently available management tools are not able to perform this reachability monitoring task in a scalable way[2]. There is a need for a monitoring system for multicast reachability both in intra-domain and inter-domain environments. While there are obviously a number of management issues, multicast reachability is the focus of this paper.

In this paper, we propose a new protocol to monitor multicast reachability within and between multicast-enabled domains. Our proposed protocol, called the Multicast Reachability Monitor (MRM) protocol, offers a new paradigm for collecting data about the availability and quality of multicast traffic carried in a network. MRM facilitates multicast network management by providing an underlying collection mechanism for data necessary to provision network management. More specifically, MRM provides network managers with a mechanism to run multicast test sessions between network devices and collect various statistics about data quality as seen by receivers. Moreover, its design enables MRM to be used in conjunction with other available management tools. From this perspective, the MRM protocol can be used as a building block for an integrated multicast network management tool set. An important component of our work is satisfying the "sufficient and necessary" condition for a new protocol. We show that other current multicast management solutions do not provide necessary functionality and that MRM is sufficient to meet our objectives for a new management protocol.

The remainder of this paper is organized as follows. Section 2 presents requirements for monitoring reachability in a multicast network. Section 3 describes the MRM protocol in detail. Section 4 discusses possible scenarios for MRM use. Section 5 compares MRM with some other related work. The paper is concluded in Section 6.

2 Issues in Monitoring Multicast Reachability

While our focus in this paper is on reachability as an instance of a management function, we believe it is necessary to comment generally on management. Because reachability monitoring will be conducted in parallel with other management functions, support for reachability monitoring should use similar mechanisms and provide a consistent interface.

In general, the goal of network management is to organize and highlight relevant information about the network including topologies, configurations, data flow, performance statistics, and *especially* anomalies. This allows people without in-depth knowledge of a network's configuration to (1) monitor its operation, (2) easily identify problems when they occur, and (3) solve problems based on the availability and presentation of relevant information.

Two types of management are relevant to our work: within an enterprise (intra-domain), and inter-domain. Under ideal circumstances, management functions for an enterprise network should be orchestrated from a NOC. The NOC provides a centralized facility to receive, process, and display network status information. Even in the inter-domain case, monitoring and management functions will be somewhat centralized. Data collection and analysis are typically coordinated at a single site. However, any corrective measures will likely be distributed, i.e. performed at remote sites. Specifically, problems will be fixed by personnel in the domain in which the problem has occurred.

With this perspective in mind, we now turn our attention to developing a high-level set of requirements for a management system. Again, our goal is to specifically deal only with reachability. However, we also consider how it might be used to support other management operations. We then describe MRM, and compare it to other existing solutions like the Simple Network Management Protocol (SNMP), etc. In our opinion, some of the important requirements for a multicast reachability monitoring architecture include the following:

- **Intra- vs Inter-Domain Support**: A multicast reachability monitoring system should support both intra-domain and inter-domain monitoring. The two different scales have potentially different functional requirements. For example, access to network devices in the intra-domain is typically unlimited and devices are under the control of the NOC. Management personnel will not only have full access to all devices but they will also be expected to solve problems by changing parameters or configurations.

  Reachability monitoring in the inter-domain is based on much more limited access to devices. In some cases, access may be so limited that devices *in* the network cannot be used for monitoring and only end hosts are available. As we will see, the potentially restricted access to information drives other requirements for the reachability monitoring architecture. In any case, the information that is made available can be very useful for confirming (a) that problems do exist, and (b) that problems are located in a remote domain. Reachability monitoring is a management function well suited for the inter-domain because it requires less detailed information from connected domains.

- **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 reachability monitoring system should adopt mechanisms to prevent unnecessary traffic on the network and excessive processing load on the management station and on the participating nodes. One particularly important scalability characteristic any monitoring system should have is the ability to control report implosion. Implosion occurs when a centralized collection site is used and the system supports "alarms", i.e. asynchronous reports of error conditions. For example, while monitoring reachability in a multicast tree, a failure of a critical link close to the source would likely cause many receivers to generate reports. Too many reports would cause implosion or at least excessive load at the collection site.

- **Security**: Security is another important issue in reachability monitoring systems. There is a set of requirements common to intra-domain and inter-domain systems alike, but there are also additional issues for inter-domain monitoring. Network devices used in intra-domain reachability monitoring tests are usually production components and should only be accessed in a controlled manner by authorized personnel. These devices can usually 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 attacks is much greater. In addition, security mechanisms need to be used to control how one domain uses devices in another domain. Even NOC personnel who are open to cooperation will want tight control over what kinds of tests local devices are requested to perform or participate in.

  Since we have made a distinction between in-the-network devices and end hosts, it is worth noting that there are different security requirements for these two classes of devices. Both should have mechanisms in place to limit access by using access control lists. Furthermore, end hosts used in inter-domain reachability monitoring tests should be protected against excessive monitoring workloads. One distinction between in-the-network devices is that end hosts may be set up by normal users (including researchers) and NOC personnel may not need to be involved. In addition, significantly more processing power and flexibility may be achieved with end hosts.

- **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. The protocol does not specify what is collected but rather specifies how data is collected. Additional specifications, called Management Information Bases (MIBs)[3], give standards for what kinds of data is collected and how it is organized. Similarly, reachability monitoring systems should support collection of additional data sets and they should support using new test data formats where possible.

- **Device Flexibility**: As briefly mentioned earlier, the reachability monitoring system should support data transmission and reception functionality in different types of devices. 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. While such diverse support is not needed for monitoring reachability, at least devices in-the-network and end hosts should be supported. End hosts provide a good end-to-end view of reachability plus possibly additional statistics about stream quality. Support in the network provides additional sources of information for fault isolation when problems are detected.

- **Multicast Independence**: In a multicast reachability monitoring system, successful communication between a monitoring coordination site and participation sites, even in the presence of faults, is important. Reachability 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 may break and the effectiveness of monitoring jeopardized. Making the point about non-reliance on multicast is important because it is a particularly effective mechanism for achieving scalability. Several existing systems provide excellent functionality but only when multicast is working properly[2].

3 The Multicast Reachability Monitor (MRM) Protocol

In this section, we describe our design for a new protocol to send and receive test multicast data streams and to collect information about the quality of the stream. Our proposed protocol, called the Multicast Reachability Monitor protocol, is designed largely based on the requirements described in Section 2 and consideration for existing systems. MRM is designed to support basic reachability monitoring plus provide "hooks" to other management systems and tools.

The functionality provided by MRM is the ability to have a centralized management station configure test multicast sessions. These sessions include who the source(s) should be, what the transmission rate should be, who the receiver(s) should be, and criteria for reporting results or asynchronous alarms. A network manager can create test scenarios that monitor traffic conditions for an almost unlimited set of scenarios. Generated traffic can be used to test basic reachability or test end-to-end capacity. Tests can be conducted in advance of an event to confirm functionality or during an event to monitor quality. Tests can be conducted frequently using a constantly changing subset of network devices. The goal in this case would be to statistically test reachability for very large networks. More detailed examples are described in Section 4.

Given that MRM has been designed in accordance with the requirements in Section 2, there is a parallel set of specific design criteria. As such, the MRM design goals specifically include the following:

- MRM should provide close to real-time detection of reachability problems. This includes some amount of fault avoidance as well as fault isolation once a problem is detected.
- MRM should have the ability to provide good network coverage. This suggests MRM functionality should be available in both routers and end hosts.
- MRM should support scalability. This includes both the ability to adequately monitor a large network, and the ability to handle the feedback generated by widespread error conditions.
- MRM needs good extensibility in terms of the protocol's ability to collect varied and complete sets of diagnostic data. The needs that exist and are designed for today should not limit the protocol's future use.
- MRM should have the ability to interact with other diagnostic tools. This includes the ability to collect information using these tools and then incorporate it into the presentation of network status. Because MRM does not provide presentation functionality, hooks need to be provided for other visualization tools[4, 5].
- MRM must be flexible enough to support use in both an intra- and inter-domain environment.
- A sufficient level of security that protects MRM-capable devices from being controlled by non-authorized personnel. Also, MRM-capable devices should be protected from being overloaded by monitoring tests by authorized users.

The remainder of this section is dedicated to describing the MRM protocol in detail. The first of two remaining sections is a description of MRM protocol features including system components, scalability mechanisms, and functional interoperability. The second section contains a description of MRM's communication primitives. Communication between MRM components is described by listing the MRM message types.

3.1 MRM Protocol Description

3.1.1 MRM Components

An MRM system consists of two types of components. First, an *MRM manager* that configures tests, collects data, and presents information. Second, *MRM agents* that source or receiver test traffic. There is one agent per device and the device can either be a router or an end host. Each agent can run two type of processes, either a *Test Sender (TS)* or a *Test Receiver (TR)*. A TS acts as a traffic source and a TR receives, processes, and reports on the traffic it receives. A single device running an MRM agent can act as a TS or TR for potentially many test sessions simultaneously. Figure 1 shows a hierarchy of the MRM terminology.

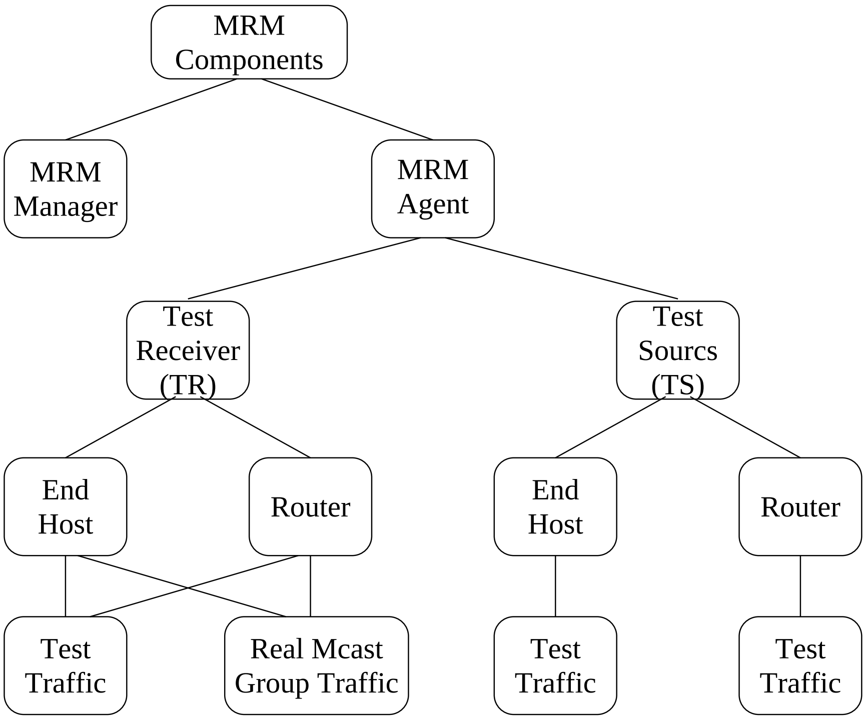

Figure 1: A hierarchy of MRM terminology.

An MRM test has four basic steps. First, the MRM manager sets up the test scenario. 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 shown in Figure 2 and described in more detail below.

1. An MRM manager instantiates a test scenario based on parameters from a network engineer. The MRM manager initiates configuration requests to the MRM agents and assigns the roles of TSs and TRs. The MRM manager informs the TSs of the quantity and duration of traffic to generate and informs the TRs of the types of reports to generate. Access rights to MRM agents can be controlled using access lists, and MRM agents can use the IP Security Authentication Header[6] with HMAC-MD5 transformation as the standard authentication algorithm[7].

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. The 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 RTPv2 packet header[8] is used. This includes both traffic generation as well as

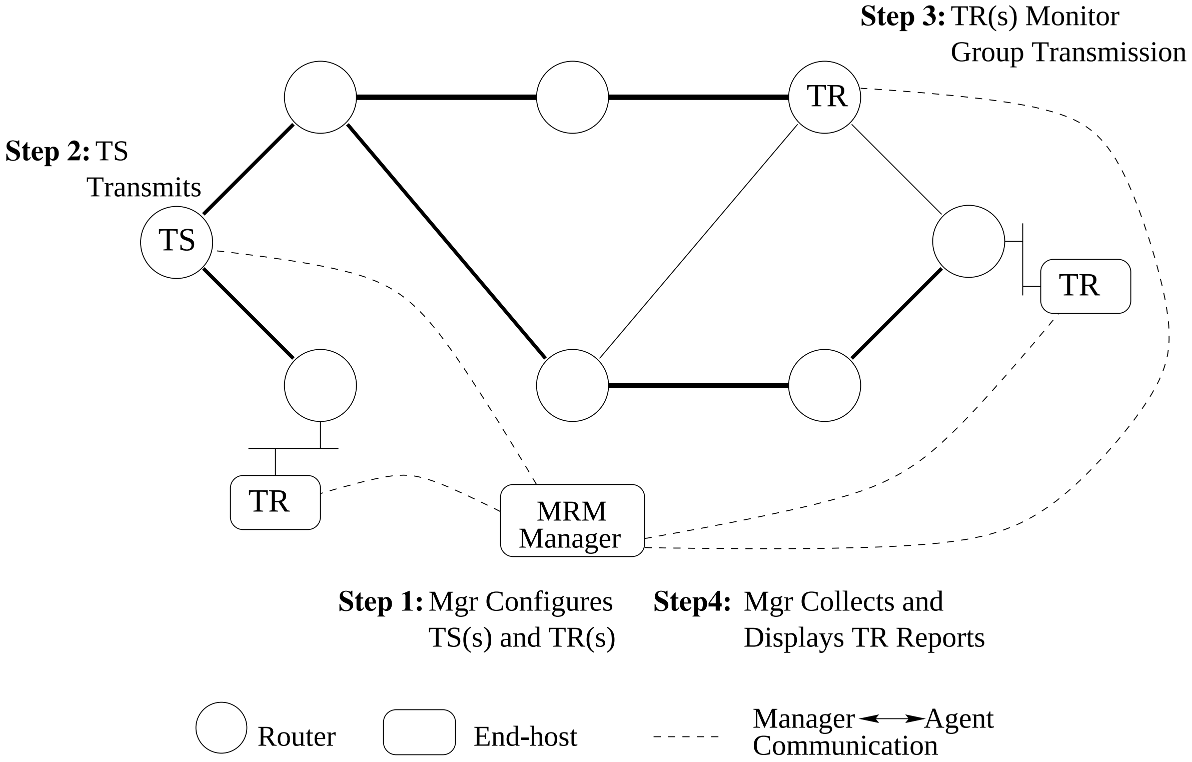

Figure 2: The architecture of an MRM system including message flow

TR status report messages. This allows re-use of existing RTP-based reception mechanisms and provides interoperability with existing RTP-based tools.

There are a number of additional data types which would prove useful including burst error patterns, long term loss statistics, one-way delay measurements, group fan-out, etc. Development of extended RTP headers is underway[9]. In some cases, MRM might even want to collect and report data by setting SNMP MIB variables. The challenge is to minimize the impact of requiring the router to collect additional data while maximizing the value of the data itself. Utilizing different capabilities in routers and end hosts, specific functions can be tailored to specific types of devices. For example, end hosts can collect and store information about the arrival of *each* packet. Our assumption is that much more refined collection and processing can be done in an end host than can be done on a router.

Finally, MRM agents need to have the ability to accept or deny incoming test requests based on their current workload. This mechanism prevents network devices from becoming overloaded by monitoring operations. This is particularly important in the inter-domain environment.

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. In the next section we describe mechanisms to achieve scalability in ways that SNMP cannot.

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[4, 5].

3.1.2 MRM Scalability

MRM suffers from potential scalability problems. 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. As with some reliable multicast protocols, the source (in this case the MRM manager) is responsible for collecting acknowledgments (reports) from many receivers (TRs). The potential for report implosion is significant, especially in the case where TRs are configured to generate reports in response to threshold violations. However, report implosion can also occur when a test is configured and very detailed statistics are requested from large numbers of receivers.

Another application in the multicast world that must deal with report implosion is RTCP. RTCP has its own scalability mechanisms (feedback back-off), but this technique is not wholly applicable to MRM. The reason is that with RTCP, the more receivers that join a group, the more infrequent the feedback. For MRM, this may not be an option. A manager may simply want a large amount of feedback, even if it increases the possibility of overload.

MRM's solution to the scalability problem is to utilize techniques from RTCP and from some reliable multicast protocols. The basic lesson that reliable multicast protocol developers have learned is that receivers can achieve reliability without having to send acknowledgments to the source for every single packet[10]. For MRM, which scalability techniques are used will depend on what the test scenario is and what the needs of the network manager are. Some of the techniques available include the following:

1. **Delayed feedback**: The MRM manager assigns a pre-determined report-delay (as part of the configuration design task) to each TR. Each TR upon detecting a fault, will randomly delay the sending of its report based on the pre-set delay period. This would allow an MRM system

to monitor networks with up to thousands of systems without unreasonable compromises in detection response times.

2. **Report suppression**: Each TR may be instructed to report the detected faults to a multicast group address using feedback techniques similar to RTCP. Other TRs who hear information similar to their own may suppress the delivery of their reports. However, one problem with suppression is that it will prevent the MRM manager from learning a complete list of receivers affected by a specific fault. Report aggregation is an alternative, but it requires additional functionality in internal MRM agents.

3. **Report aggregation**: Report aggregation is a technique used successfully by reliable multicast protocols. The aggregation functionality can be provided by internal network devices or by specially elected end hosts. Report aggregation is a potentially powerful tool for summarizing relevant information at branch points in a multicast tree.

4. **Non-realtime feedback**: This is an alternative to event triggered reports. Non-realtime feedback implies that an MRM agent stores reports for an extended period of time. If a significant event negatively affects many members in a large group, some reports can be sent immediately but most could be stored on the MRM agent host machine. When the MRM manager is ready to collect the data it will send out poll messages–possibly even using an ftp-style protocol if the log is large.

5. **Beacon messages**: MRM has a protocol service that allows an MRM components to communicate via multicast. The utility of this mechanism is somewhat limited because MRM cannot rely on multicast as a reliable means of communication. Beacon messages are described further in Section 3.2.

3.1.3 MRM Interoperability

MRM was designed, not to replace existing management tools and systems, but to work in conjunction with them. From one point-of-view, an MRM manager is responsible for generating and collecting MRM reports. From another, MRM may also be used by any other tool to help the network manager understand multicast traffic behavior. Some examples of interoperability include the following:

- An MRM manager may want to collect data from other management tools, e.g. *mtrace*. The MRM manager would send messages to MRM agents asking them to run tests using other tools on behalf of the manager.
- MRM may be considered a tool and incorporated into a larger management frameworks. In particular, work is underway to incorporate MRM into a tool called *mmon*[11] which is part of OpenView. MRM could also be included in the Globally Distributed Troubleshooting (GDT) protocol[12].
- MRM could have an SNMP MIB and populate it based on test results from a scenario configured by the MRM manager. This provides yet another mechanism for making MRM data available to network management personnel.

- A management application could provide a layer of abstraction for the MRM protocol. Instead of requiring network engineers to configure scenarios, the management application would generate generic scenarios. These scenarios would be customizable and could be stored and run using the management application.
- As mentioned before, MRM could be used as an input source to end-user tools that perform visualization and debugging assistance[4, 5].

3.2 MRM Messages

MRM functionality is based on the ability of an MRM manager to configure and interact with TSs and TRs. This section describes the types of messages that flow during a typical MRM scenario.

Test_Sender_Requests (TSRs). TSRs cause an MRM agent to begin sourcing packets according to the parameters in the TSR packet. TSR messages are sent using unicast UDP with acknowledgments. Additional soft-state updates may be carried in beacon messages (see below). A TSR packet contains the following information:

- Duration of the test scenario
- Inter-packet interval
- Length of each test packet
- Format of test packets, e.g. RTP/UDP, UDP, or some other format
- Multicast address for the test group

Based on this information, a TS will start sending periodic test packets. The length and inter-packet interval can be used in combination to generate streams of varying bandwidth.

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 become a TRR or it can be a request to an existing TRR to return a status report. In either case, a TRR packet includes the following information:

- The address of the group to be monitored
- A list of source addresses to record reception quality information for

- A description of the threshold used to trigger fault reports/alarms
- The maximum delay to wait before generating fault reports. Typically the arrival of the first test packet will start the collection process. However, if no packet arrives the TR needs a catalyst to recognize that the test has started and no packets have arrived.
- An interval over which to chose a random delay between when a fault is observed and when a fault report is sent (backoff to prevent implosion)
- The type of report to be returned, e.g. RTCP or some other format[9]

Test_Receiver_Status_Reports (TRSRs). These reports are sent by the TRs to the MRM manager, either in response to a status request or because of a threshold trigger. The initial design is for status reports to use the RTP "receiver report (RR)" packet format. MRM is designed to be extensible and to support more detailed reports. Several extended report headers are currently in development but their format is not part of the MRM protocol[9].

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 either a generic multicast group or a group specific to a particular test scenario. The decision on which to use and why is explained below. All TSs and TRs join this multicast group and listen for beacon messages. 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 and to calculate reception quality from the MRM manager. 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 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. Therefore, even if beacon messages are made reliable (which they are not), 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. However, the re-assert mechanism is limited to a small-scale testing environment because a

pre-configured multicast address must be used for the beacon group. In a large-scale test there will likely be too much TS and TR state to send out in a beacon. We provide the option of using a different beacon address per scenario but then a network device will not be able to recover this address upon re-start.

An original protocol design component was to allow TSRs, TRRs, fault reports, or status reports to be sent via multicast. This created a dependency on multicast which could affect MRM operation. The solution is to make the use of beacon messages by the MRM manager optional. However, MRM agents are required to respond to any multicast messages received. So while beacon messages provide robustness they do not provide any critical functionality.

4 MRM Usage Scenarios

The basic concept of MRM is to allow a network manager to monitor multicast reachability and stream quality at one or more receivers. The traffic can either be real traffic or can be artificially generated trace packets. By allowing a network manager to arbitrarily place sources and receivers throughout the network, important flexibility in monitoring functions can be supported. A network manager can check the quality of the network links between any potential source and any potential set of receivers. This check can be performed for existing groups during a real session or can be performed in advance of an event using test traffic. The following four scenarios demonstrate more fully the various ways in which MRM can be used.

Pre-Event Testing. One of the best examples of how MRM can be used is as a deployment verification test tool for on-location broadcasts. Examples include meetings or conferences, especially those that periodically take place in venues around the world. Preceding the meeting, network support staff install a terminal room and establish network connectivity. In some cases, setup activities occur only days before the event. Verifying that multicast routing is working both into and out of meeting rooms is a challenge. Ensuring that multicast can be received and is of acceptable quality beyond the venue location is difficult. MRM could be used to (1) establish sources in the local network, and (2) establish randomly located receivers around the world. Today this is typically done by having one of the network support staff set up a test session and solicit feedback from users. MRM would ease the process of testing deployment especially if there were a publicly accessible/usable set of MRM agents.

Classic Fault Isolation. A second scenario well suited for MRM is classic fault isolation. Like unicast routing, multicast routing problems can be very difficult to debug. But unlike unicast routing, the additional complexities of providing one-to-many delivery can introduce problems that are difficult to find. To date, a significant number of strategies, tools, and techniques have been developed, built, and proposed[13, 14]. However, these attempts generally require a significant level of multicast routing expertise and experience–characteristics not always found among NOC personnel. As a result, MRM is designed to offer a layer of abstraction between multicast reachability monitoring and the intricacies of multicast routing. Pre-configured tests can be used to generically test network reachability, and any problems can be further isolated using additional tools.

Session Monitoring. The two previous scenarios follow logically, from verifying multicast connectivity to isolating any potential faults. The next scenario is monitoring of existing, active sessions. Such groups will have a well-known multicast address, and might be exchanging group membership information via RTCP reports or some other out-of-band mechanism. While RTCP is a common solution in widespread use today, applications are increasingly not implementing group-wide reporting mechanisms. In this case, other steps need to be taken to monitor quality. Even if RTCP is used, the current standard does not allow flexibility in the amount of control traffic allowed[8]. Furthermore, RTCP, especially the version built into application tools, may not support all of the monitoring functionality desired. Specific monitoring tools, which only support debugging functions and not application layer functions like image decoding, may be the only option for reachability monitoring. Instead of using yet another technique/solution/tool, MRM provides a consistent management model. MRM also provides a great deal of flexibility and extensibility; all of which can be controlled from the NOC.

Fault Logging. In the case when session monitoring identifies the existence of a fault, a range of logging functions may be required. At one extreme, the MRM manager may simply need to be alerted when faults occur so that appropriate investigative measures can be taken. At the other extreme, service contracts may depend on the provision of service with certain guarantees. Any outage might need to be closely tracked. These two extremes again demonstrate the need for MRM to be flexible. In particular, when faults need to be closely monitored and logged, a wide-scale outage may itself cause a heavy load on the network. While scalability is an important goal, the tradeoff is that less information about the problem is received. MRM supports scalability, but the issue of when to make monitoring functions scalable and when to collect full statistics is an important attribute.

5 Related Work

In this section, we look at four alternative mechanisms to perform reachability monitoring task: sdr-monitor, SNMP, the NIMI Project, and GDT. In reality, SNMP and GDT actually provide complementary functionality. However, we treat all four as alternatives as much as possible in order to identify their strengths and weaknesses when compared to MRM. For each alternative, we explain how it could be used to perform reachability monitoring.

Sdr-monitor. Sdr-monitor is a user-level tool for monitoring multicast reachability on a global scale[15]. It uses *sdr* session announcements as a heartbeat mechanism for measuring reachability. Sdr-monitor has a number of participants who periodically report their cached session announcements to the sdr-monitor data collection site. An sdr-monitor manager then uses these reports to build a web page displaying which sessions are visible to which participants. A session that is not visible to a particular sdr-monitor participant is assumed to be an indication of a reachability problem.

Even though sdr-monitor provides useful information about multicast reachability in the global infrastructure, it has a number of problems inherent to its underlying data collection mechanism. Using the sdr-monitor tool, we cannot monitor multicast reachability between two specifically targeted sites. The monitoring that takes places is completely dependent on the currently participating and session announcing sites. Participants can join and leave the monitoring effort as they wish. In addition, sites start and stop sourcing session announcements as they wish. In summary, sdr-monitor provides reachability information based on the existing session announcement mechanism. Even though this monitoring is useful in a general sense, it does not provide a mechanism to perform monitoring in a flexible way. In addition, sdr-monitor provides only binary information (reachable, or not reachable) and cannot provide any information about the quality of data reception, e.g. loss, jitter or delay information.

SNMP. SNMP is a standard protocol commonly used for network management by NOC personnel. It provides a simple mechanism for network managers to collect statistics from and affect state in managed network devices. The type of management data gathered in network devices are defined by MIBs. Reachability monitoring task can be performed using SNMP. The TS and TR functionality can be implemented as MIBs and network devices can be configured to participate in a monitoring test scenario using SNMP commands. Even though this looks like a practical solution to our monitoring task, there are problems that make SNMP unsuitable. SNMP uses a password mechanism in order to prevent unauthorized accesses to SNMP data in managed network devices. Theoretically, managed devices can be configured to allow others to access SNMP data in the devices (using community strings). However, traditionally most network managers do not allow non-NOC personnel to access SNMP information in their devices. They configure network devices to only accept SNMP requests from their own NOC personnel. This makes SNMP an intra-domain management system only. Although this mechanism provides a perfect solution for reachability monitoring within an autonomous system, it does not support wide-scale reachability monitoring tasks. We believe support for inter-domain reachability monitoring is an important task–even for NOC personnel.

The NIMI Project. Another alternative platform that can be used to provide multicast reachability monitoring is the National Internet Measurement Infrastructure (NIMI) project[16, 17]. NIMI has been developed based on the need for a global Internet measurement infrastructure. NIMI provides an infrastructure in which a collection of measurement probes cooperate to measure the properties of Internet paths and clouds. NIMI is an active measurement system and is expected to generate its own test traffic. NIMI uses multiple NIMI daemons at end-points as the set of measurement tools. From a reachability monitoring perspective, TS and TR functionality can be implemented as NIMI modules that can be placed and operated within NIMI-enabled end hosts. Even though this technique provides a very viable solution for end hosts, NIMI is not supported in routers. Even though NIMI provides a mechanism to perform end host monitoring, we still would need MRM for router-based test scenarios. Furthermore, NIMI is designed almost entirely for use by researchers. This is little functionality that would be useful for management of commercial network services.

GDT. GDT provides a mechanism to detect and report network problems across administrative domains. In GDT, each domain has a number of expert modules which has one or more 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. GDT is designed as an application-layer inter-domain debugging coordination tool.

Any entity within a network 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 the problem is believed to exist in another domain, the hypotheses 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. From a reachability monitoring perspective, GDT can be considered as a higher level management architecture which expects to use other mechanisms (like MRM) to detect reachability problems.

Using the set of requirements described in Section 2 we now present Table 1–a summary of how well the various management systems support MRM-style multicast reachability monitoring.

6 Conclusions

In this paper we have attempted to address the problem of a lack of management tools for multicast, specifically for the function of reachability monitoring. This deficiency exists because the tools in use today do not meet the needs of traditional network management. The result is that there is a significant need for easy-to-understand and easy-to-use multicast management tools. We proposed a protocol that enables network managers to monitor the reachability and quality of multicast traffic. The key to our protocol is its ability to configure test sources and receivers and create a wide range of multicast group configurations. This flexibility allows a manager to run a variety of tests and collect statistics about multicast data transmission. Based on this information, a network manager can use other multicast management tools to detect and correct multicast related problems. From this perspective, our protocol can be used as a building block for an integrated multicast management system. In this paper we explained components of our protocol and described various scenarios for its use. In addition, we discussed other management systems that offer minimal reachability monitoring functionality.

Intra- and Inter-Domain Support	
sdr-monitor	supports intra- and inter-domain monitoring
NIMI	supports intra- and inter-domain monitoring
SNMP	typically supports only intra-domain monitoring
GDT	supports primarily inter-domain communication
MRM	supports intra- and inter-domain monitoring
Scalability	
sdr-monitor	does not scale, heavy workload with many participants
NIMI	can employ mechanisms to prevent report implosion
SNMP	limited scalability–alarm report filtering
GDT	minor bottleneck for handling problems report to modules
MRM	can employ multiple mechanisms to avoid bottlenecks
Security	
sdr-monitor	no security provisions
NIMI	uses authentication and passwords to prevent abuses
SNMP	strict security model limits access, especially in inter-domain case
GDT	hooks to provide security
MRM	authentication to control access and workload
Extensibility	
sdr-monitor	limited, depends on session announcement mechanism
NIMI	possible to add new modules
SNMP	only through definition of new MIBs
GDT	possible to add new expert modules
MRM	possible to support new collection types, especially in end hosts
Device Flexibility	
sdr-monitor	end host support only
NIMI	end host support only
SNMP	most network devices
GDT	end host support only
MRM	end host and router support

Table 1: Summary of how well existing systems meet reachability monitoring requirements.

References

[1] C. Diot, B. Lyles, B. Levine, and H. Kassem, "Requirements for the definition of new IP-multicast services," *IEEE Network*, January/February 2000.

[2] K. Almeroth, "Managing IP multicast traffic: A first look at the issues, tools, and challenges." IP Multicast Initiative White Paper, August 1999.

[3] 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.

[4] P. Rajvaidya and K. Almeroth, "A scalable architecture for monitoring and visualizing multicast statistics," tech. rep., University of California–Santa Barbara, 2000.

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

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

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

[8] 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.

[9] 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.

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

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

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

[13] D. Thaler and B. Aboba, "Multicast debugging handbook." Internet Engineering Task Force (IETF), draft-ietf-mboned-mdh-*.txt, March 1997.

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

[15] K. Sarac and K. Almeroth, "Sdr-monitor: A global session monitoring tool," tech. rep., University of California–Santa Barbara, March 2000. (submitted).

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

[17] *National Internet Measurement Infrastructure (NIMI) Project.* http://www.ncne.nlanr.net/nimi/.