Providing Scalable Many-to-One Feedback in Multicast Reachability Monitoring Systems

Kamil Saraç and Kevin C. Almeroth

Department of Computer Science
University of California
Santa Barbara, CA 93106
{ksarac, almeroth}@cs.ucsb.edu

Abstract. With the deployment of native multicast in commercial networks, multicast is getting closer to becoming a ubiquitous service in the Internet. The success of this deployment largely depends on the availability of good management tools and systems. In this paper, we focus on reachability monitoring as an important multicast management task. First, we provide a general architecture for multicast reachability monitoring systems and focus on three critical functions: agent configuration, monitoring and feedback collection. For each component, we provide a number of alternative approaches to implement the required functionality and discuss their advantages and disadvantages. Then, we focus on the feedback collection component. To a large extent, it determines the complexity and the overhead of a monitoring system. We compare a number of alternative approaches for feedback collection using simulations and make suggestions on when to use each. We expect our work to provide insight into the issues and considerations in designing and developing multicast reachability monitoring systems.

1 Introduction

With the deployment of native multicast in commercial networks, multicast is getting closer to becoming a ubiquitous service in the Internet. However, before multicast can be used as a revenue-generating service, its robust and flawless operation needs to be established across both the intra- and inter-domains. This requires availability of good management tools to help network administrators configure and maintain multicast functionality within and between multicast enabled domains.

While unicast management is well established and provides good support for robust network operation, there exists a need for good multicast management tools and systems. The difference between *unicast* management and *multicast* management derives mainly from the simple fact that multicast traffic can be destined to multiple receivers. Multicast data sent to multiple receivers travels along a logical tree that is dynamically built on top of the underlying network infrastructure. The construction and maintenance of this tree topology is automatically done by network devices (e.g. routers) depending on the joins and leaves of multicast session receivers. It is this dynamic behavior that makes multicast more difficult to manage and monitor compared to unicast services in the network.

The management function that we focus on in this paper is reachability monitoring. Reachability monitoring is one of the most important yet one of the most difficult multicast management tasks. Reachability monitoring enables a network operator to test availability and quality of multicast service in a domain. In this paper, we provide a general architecture for multicast reachability monitoring systems. First, we identify the functional components of this architecture and then the steps to configuring a reachability test. Next, we provide a number of alternative approaches to implementing these steps. The variations in the implementation of these steps are mainly due to the type and scope of different monitoring tasks. In this respect, we provide some guidelines about when to use each alternative implementation. We believe that this analysis is useful as guidance for reachability monitoring system designers and developers.

The rest of this paper is organized as follows. Section 2 motivates the multicast reachability monitoring problem. Section 3 provides a general architecture for reachability monitoring systems. Section 4 gives a detailed discussion on alternative approaches for implementing the functional steps of monitoring systems. In Section 5, we provide simulation-based comparisons for different feedback collection approaches and present guidelines for their applications in Section 6. The paper concludes in Section 7.

2 Motivation

From network management point-of-view, successfully deploying multicast requires the ability to instill confidence that the network is working in a correct and predictable manner. This requires mechanisms to monitor and verify successful multicast data transmission within and between multicast-enabled domains. For a globally-scoped multicast application, a number of potential receivers may be located in other domains and the delivery of data to these receivers may be affected by reachability. Network operators must have the ability to ensure multicast reachability to all potential receivers. There are two properties that make this difficult: (1) the dynamic nature of multicast trees, and (2) anonymity of group receivers.

The goal of reachability monitoring is to verify and help maintain the robust operation of multicast. Reachability monitoring within a domain is most effective when both routers and end hosts are used as monitoring agents. Monitoring agents can be configured to source test data to a multicast group and/or join and collect data reception statistics about the group. This information can then be used by the Network Operations Center (NOC) personnel for network management purposes. In general, management personnel not only have full access to all devices in the network but they are also expected to solve problems by changing parameters or configurations. Even with this capability, NOC personnel cannot control or easily monitor networks outside their own domain. Therefore, intra-domain reachability monitoring may not be enough to maintain a robust multicast service. Availability of data from sources external to the local network depends on proper multicast operation in and between other domains. This requires the ability to perform inter-domain reachability monitoring tests. These tests require access to agents located in remote domains. In general, due to security and privacy issues, network operators are not willing to share their network resources with others. Therefore, inter-domain monitoring test scenarios are generally limited to using end-host-based monitoring agents. In any case, the information that is made available can be very useful for confirming that (1) problems do exist, and (2) problems are located in a remote domain.

3 An Architecture for Multicast Reachability Monitoring

In this section we present a general architecture for multicast reachability monitoring systems. In this architecture, we focus on the basic system components and their functionality. In general, a multicast reachability monitoring system has a *manager* component and one or more monitoring *agent* components. The manager is a software component that can either be a stand alone program or can be part of a complex management system that interacts with the other modules in performing network management. Monitoring agent functionality can be implemented both in the routers and in end-hosts. Figure 1 shows an example of this architecture. In this figure, the manager uses both router and end-host agents in the intra-domain and uses end-host agents in inter-domain. In either case, the overall monitoring task can be divided into three steps: (1) the manager configures agents for a monitoring task, (2) agents perform monitoring and collect statistics, and (3) the manager collects statistics from the agents. In the next section, we discuss these steps in more detail.

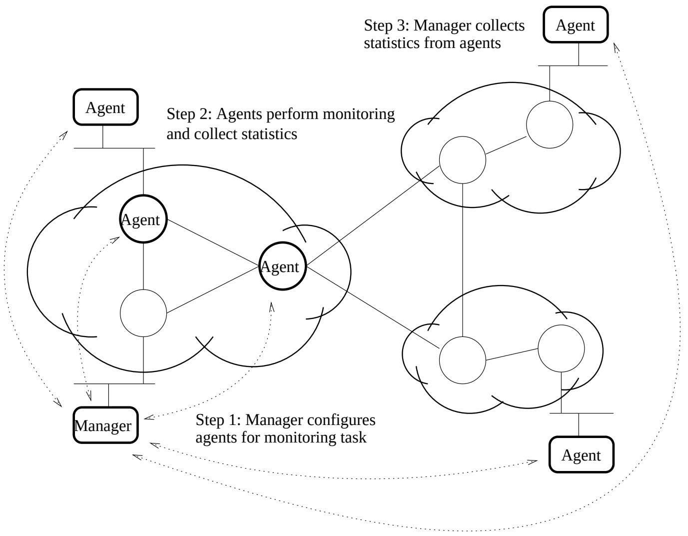

Fig. 1. A general case for intra- and inter-domain reachability monitoring.

4 Reachability Monitoring Steps

In this section, we discuss the steps involved in performing reachability monitoring. For each step, we briefly discuss the main functionality. Then, we provide alternative approaches to implement this required functionality. When appropriate, we compare these alternatives based on various criteria such as scalability, functional overhead and security.

4.1 Step I: Agent Configuration

In this step, the manager configures the agents for a particular monitoring task by sending necessary configuration parameters to the agents. These include the session address, duration of monitoring, packet encoding format, types of feedback reports to collect, etc. The type of communication used between the manager and the agents is an important consideration. Depending on the scale and complexity of the system, this communication may be as simple as a UDP-based message exchange (with or without reliability) or it may be as complex as some type of authenticated and encrypted message exchange. Important issues in this step are (1) to protect monitoring agents from un-authorized users and (2) to protect monitoring agents from being overloaded by monitoring requests.

Conducting reachability monitoring introduces both network overhead and processing overhead. Preventing the overload of router-based monitoring agents is important as it may interfere with the overall performance of the router in forwarding traffic. Therefore, the agent configuration step should use an appropriate mechanism to control access to monitoring agents. In this paper, we discuss two alternative approaches for agent configuration communication: the Simple Network Management Protocol (SNMP) and an IP Security (IPsec) based authentication mechanism:

- *SNMP-based approach.* SNMP provides a standard approach to communicate configuration or control information between a management station and monitoring agents. Traditionally, SNMP is used to access network devices to exchange management information[1]. It is a common piece in many network management systems. Most NOC personnel are comfortable with SNMP and most are familiar with SNMP-based systems. However, SNMP is not without its deficiencies. Until recently, SNMP has been primarily used for monitoring and performance management within domains. The current protocol version, SNMPv2, provides poor support for inter-domain network management tasks and suffers from lack of scalability and security. To a large extent, these problems are being addressed by two threads of effort. First, there is effort in the Internet Engineering Task Force (IETF) to develop a SNMPv3 protocol with better support for security and device configuration[2]. Second, there are research efforts looking to improve SNMP's distributed management capabilities—primarily by adding scalability[3]. We believe that with these modifications, SNMP can provide solid vehicle for control/configuration communication for reachability monitoring.

– *IPsec-based approach.* This approach refers to cryptographically strong but non-SNMP-based techniques. As an example, the Multicast Reachability Monitor (MRM) protocol uses *access control lists* and the IPsec Authentication Header (AH) mechanism to control accesses to monitoring agents[4]. Agent platforms keep a list of authorized sites for agent use and agent configuration messages of these sites use AH mechanism to protect against spoofing attacks. Another example that uses a solution other than SNMP is the National Internet Measurement Infrastructure (NIMI) project. In NIMI, monitoring platforms are protected by a password mechanism and (AH-based) authenticated message exchanges[5].

The second issue for agents is to protect them from being overloaded by excessive simultaneous monitoring requests. This is necessary because each monitoring task introduces network overhead (in terms of sourcing or receiving multicast data streams) and processing overhead (in terms of producing statistical information). The management system should be aware of the agents' load and avoid them. However, because end-host-based agents can be used by multiple managers simultaneously, the agents also need to protect themselves. Monitoring agents can regulate their overhead by limiting their network bandwidth usage in monitoring tasks and by limiting the maximum number of test scenarios that they participate in simultaneously. When an agent receives a new test request that it can not satisfy, it should deny the request and inform the requester about its decision and its reason. This protects agents from being overloaded by requests of authorized users.

4.2 Step II: Reachability Monitoring

In this step, agents perform the actual monitoring and collect data about the results. The specific functions performed by monitoring agents depend on various factors such as the *type* of the monitoring task (active vs. passive), agent platform capability (router vs. end-hosts) and granularity of collected data. In general, monitoring agent functionality can be divided into two parts: *test sender* and *test receiver*. A *test sender* is configured to source multicast data to a test multicast group address. A *test receiver* is configured to join and monitor data reception quality (either for a test session or for a real session).

Monitoring tasks can be divided into two types: *active* monitoring and *passive* monitoring. In active monitoring, network operators create a test multicast session among a number of agents. Some agents are configured to function as test senders and some agents are configured to function as test receivers. An advantage of active monitoring is that test receivers know beforehand all the important parameters about characteristics of the data stream (e.g. starting time, duration, delay interval between data packets). This information usually simplifies the computation of statistics about the session. In addition, network operators can create and run various type of monitoring tests, e.g. packet bursts, constant bit rate, or single packet transmissions. These tests are useful to verify reachability after an initial deployment or periodically to confirm proper operation. The disadvantage of these tests is that they create additional traffic in the network.

In passive monitoring, network operators configure monitoring agents to join and monitor an actual, ongoing multicast session. One advantage of passive monitoring is that it does not introduce additional traffic in the network. However, the disadvantage is that passive monitoring requires the agents to be able to interpret application layer data encoding schemes. This is necessary because the agent needs to track losses, duplicates and out-of-order packets. In addition, the monitoring depends on an active source which may not always be available or transmitting a stream with the desired characteristics.

Agent functionality can have different complexity levels and capabilities depending on the platform. Router-based agents are likely to only be able to provide a minimum support for reachability monitoring tasks. This is because routers are mainly used as production network components and any type of secondary tasks should add only minimum overhead. This requirement puts a limitation on the types of statistics that can be collected by router-based agents. As an example, MRM uses the Realtime Transport Protocol (RTP) to provide a minimum set of feedback information: packet loss, jitter and delay information[4, 6].

Compared to network devices, end-host-based agents can support the collection of more detailed statistical information. In general, end-host-based reachability monitoring systems are more extensible and can provide more flexibility for various types of monitoring tasks. For example, there are research studies that use end-host-based multicast reachability monitoring tests to collect per packet loss information and use this information to estimate loss characteristics of individual network links[7]. In a related joint study, we proposed a number of extensions to the RTP reporting mechanism[8] that could provide a standard model to collect more detailed information about multicast data reception quality. Even though end-host-based agents are better than router-based agents in almost every way, routers provide a view from inside the network that is hard to obtain otherwise.

4.3 Step III: Feedback Report Collection

In this step, agents send their feedback reports back to the manager site. The reporting mechanism can be divided into two types: off-line reporting and real-time reporting. In off-line reporting, agents store the statistics that they generate during the monitoring step and then send them to the manager at a later time. Off-line reporting requires local storage and therefore it is more suitable for end-host-based agents. Real-time reporting can be divided into two parts: polling-based reporting and alarm reporting. In polling-based reporting, agents create and send a feedback report based on an explicit request by the manager. In alarm reporting, agents create and send feedback reports based on detecting a threshold violation event. The alarm reporting mechanism has potential scalability problems. In a monitoring task where a large number of agents are configured to send alarm reports back to the manager site, a threshold violation close to the session source can cause a large number of agents to create and send feedback reports to the manager. If not controlled, these feedback reports can create scalability problems both in the network and at the manager site.

Depending on the type of test scenario, a number of different approaches can be used for alarm report collection. In this paper, we discuss four different approaches: (1) plain reporting, (2) report suppression, (3) delayed reporting and (4) probabilistic reporting. Each of these alternatives have different characteristics in terms of processing overhead, scalability and information granularity. We describe each of these alternatives below:

1. *Plain reporting.* This is the most straightforward reporting technique. We use this technique as the baseline in our analysis. In plain reporting, agents send their feedback reports directly to the manager site on detecting a threshold violation. This approach has the most potential for causing scalability problems. Depending on the size of the monitoring scenario, feedback reports may cause report implosion at the manager site.
2. *Report suppression.* In this approach monitoring agents cooperate with each other to minimize the number of feedback reports that reach the manager site. Similar approaches have been used in various reliable multicast routing protocols. Walz and Levine use a version of this approach in the Hierarchical Passive Multicast Monitor[9] (HPMM). In report suppression, agents do not send their feedback reports directly to the manager site. Instead they use a reporting hierarchy. In this hierarchy, there exists a number of nodes performing feedback suppression. Monitoring agents are configured to send their reports to appropriate suppression nodes. A suppression node can be either an agent participating in the monitoring task or an end-host specially elected for this purpose. Based on the type of suppression node, the suppression functionality may have different implementations. Suppression functionality implemented in routers should have minimum processing overhead and therefore should use a simple method for suppression. On the other hand, end-host-based suppressors can perform more complex functions. In general, the report suppression approach aims to inform the manager site about the existence and (approximate) location of a threshold violation event. However, it may not inform the manager site about which individual agents are affected by the event. This information can easily be obtained by the manager provided that it knows the monitoring session tree topology.
3. *Delayed reporting.* In this approach, on detecting a threshold violation, agents do not send their feedback reports immediately to the manager but delay them for a period of time. Delayed feedback techniques have been used in several other multicast protocols. For example, RTP uses a dynamic algorithm to compute an interval during which session receivers collect statistics about data transmission quality of the session. At the end of the interval, group members send this information back to the group. In reachability monitoring, agents select a random delay period from a given maximum wait interval. The maximum wait interval is assigned by the manager as part of the agent configuration. This approach relaxes the real-time requirement of feedback reporting and provides more scalability in the report collection

mechanism. That is, scalability is gained by extending the time interval for agent feedback reports so that they do not cause report implosion problems at the manager site. This technique is best used when there are infrequent threshold violations. On the other hand, in the case of frequent threshold violations, the scalability gain of this approach may deteriorate if several reports are generated with overlapping report delays.

4. *Probabilistic reporting.* In this approach, monitoring agents send their reports based on some probability. The reporting probability is assigned by the manager during agent configuration. In the probabilistic reporting approach, scalability is gained at the expense of potentially losing some feedback information (when the reporting probability is less than one). Therefore, this approach is only useful for collecting an approximation of the reachability status of a multicast network. A similar approach was used by Ammar to address scalability issues in distributed computing applications that require many-to-one response collection[10].

In the above list, we briefly discussed alternative approaches for alarm report collection. We pointed out the differences between these approaches in terms of their scalability characteristics and feedback information granularity. In addition to these characteristics, the processing overhead that these different approaches put on monitoring agents is also an important consideration. In terms of processing overhead, test receivers have similar behavior in all of these approaches. Mainly, they join and monitor a session; on detecting a threshold violation, they create feedback reports and send them to the appropriate destination(s). In the delayed approach, there is additional functionality that requires test receivers to keep timers to delay transmission of their reports. Also, in the suppression-based approach test receivers may have to perform suppression in addition to monitoring. In the next section, we compare the performance of these approaches using simulations.

5 Simulations

The feedback reporting mechanism used in a reachability monitoring system has an important effect on the overall performance of the system. In this section, we quantitatively compare the overhead of the feedback reporting approaches that we discussed in the previous section. We compare these approaches based on their messaging overhead at the manager site. In our comparisons, we use two metrics: (1) the number of feedback reports reaching the manager site during a 100 second simulation and (2) the peak number of feedback reports reaching the manager site in a small time interval. For the second metric, we collect data for 0.1, 0.5 and 1.0 second intervals. In this paper, we present results for the 1.0 second interval and the results from the other intervals exhibit similar behavior.

In the simulations, we use topologies with different sizes and different threshold violation values. We generate random flat graphs for 100, 200, 300, 400 and 500 node topologies using the Georgia Tech Internet Topology Modeler (GT-ITM)[11]. In these topologies, we identify one of the nodes as the *test sender* and all the others as *test receivers*. Then, we assign loss probabilities for each link in the network. For loss assignment, we use the guidelines presented in Yajnik et. al.[12]. According to this approach, most of the internal links are assigned small loss probabilities and a few edge (or close to the edge) links are assigned larger loss probabilities. The last parameter in our simulations is the threshold value necessary to generate an alarm report. We use loss-based threshold values with >0% and 10% losses. Briefly considering the effects of our choices for the parameter ranges, system behavior outside of these ranges are relatively intuitive. In general, larger graph sizes mean more agents and therefore more feedback reports for a threshold violation. More losses in the backbone means more synchronized feedback reports for the threshold violation. On the other hand, higher threshold values mean less violations and therefore less feedback reports in the system.

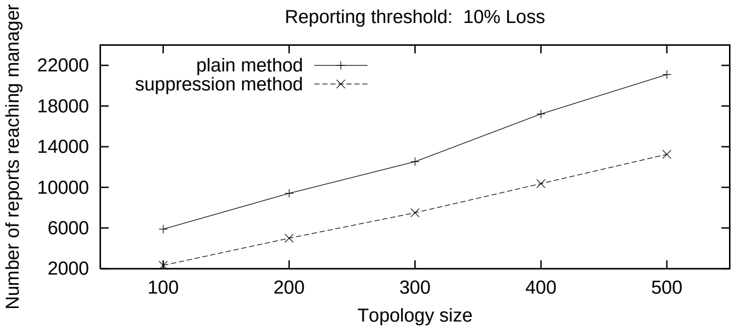

Fig. 2. Plain vs Suppression: Reports reaching manager during 100 sec simulation.

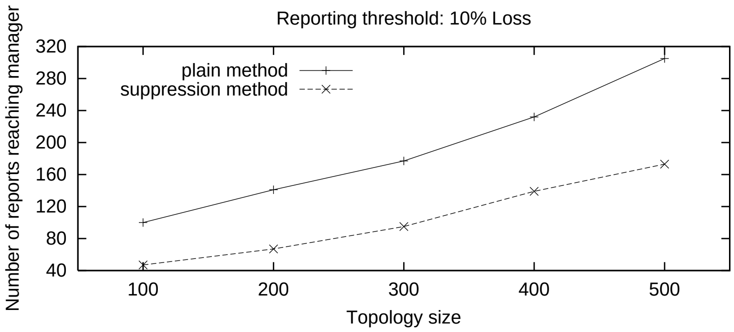

Fig. 3. Plain vs Suppression: Maximum reports reaching manager in 1 sec interval.

The first set of simulations are run to compare the plain report collection approach and the suppression based report collection approach. These two approaches have a common property in that the manager site receives feedback information for *all* threshold violation events. In other words, these approaches do not lose any information during feedback reporting. As we mentioned above, the suppression functionality can have different implementations. We implement a simple suppression function at agent nodes. According to this implementation, when an agent detects a threshold violation event, it will create and send a feedback report to its upstream neighbor. If and when this report reaches the agent at the root of the tree, this agent sends it to the manager. In our simulations, the root agent is always the manager node and at the same time it is the *test sender* of the session. After sending a report, an agent A will receive feedback reports from its downstream neighbors. This is because, any threshold violation that A detects should also be detected by the agents below it. On receiving a report from a downstream neighbor B, A will forward this report if the report is for another threshold violation event that has occurred between A and B. Otherwise, A will suppress this report. Figure 2 compares the plain and the suppression based approaches in terms of the overall number of reports that reach the manager site during a 100 second monitoring interval. The differences in the number of reports between the two lines show the saving of the suppression-based feedback collection technique. According to the figure, in this particular monitoring scenario, suppression-based feedback collection provides around 50% savings in the number of reports reaching the manager site. This saving is gained at the expense of having suppression functionality at every monitoring agent. Figure 3 compares these two approaches based on the maximum number of feedback reports reaching the manager site within a one second interval. These results present the peak number of reports that the manager must process in one second. Figure 3 shows that the maximum messaging overhead of plain feedback reporting is two times more than that of suppression-based approach. These results indicate that suppression-based feedback collection approach provides more scalability for feedback report collection without introducing any loss of feedback information.

In the next set of simulations, we compare the plain report collection approach with the delayed feedback collection approach. In this set of simulations, we use >0% loss threshold value and configure the test sender to introduce packet losses by not sending randomly selected data packets to the group address. This causes all the agents to detect threshold violation events and therefore to send feedback reports to the manager. In the simulations, we use 2, 4, 6, 8 and 10 second delay intervals. The actual delay values assigned to agents are randomly selected within these intervals. Figure 4 shows the results of these simulations. In this figure, we see that delayed feedback collection approach loses some feedback reports. According to this figure, as the delay period increases, it introduces more feedback losses into the system. Information loss can be avoided by keeping track of pending feedback reports and sending them on their delay expiration. But this introduces extra overhead into the system as agents need to keep track of all pending reports. Figure 5 shows the maximum number of feedback reports reaching the manager site within one second interval. According to this figure, the messaging overhead at the manager site goes down as we increase the delay period. In the figure, the 2 sec delayed case performs close to the plain method.

We believe that this behavior is due to the randomization in assigning delay values to monitoring agents. These results indicate that in case of infrequent losses, the delayed feedback reporting approach provides more feedback scalability in the sense that the manager does not have a report implosion problem. On the other hand, it may cause feedback information losses depending on the frequency of threshold violations.

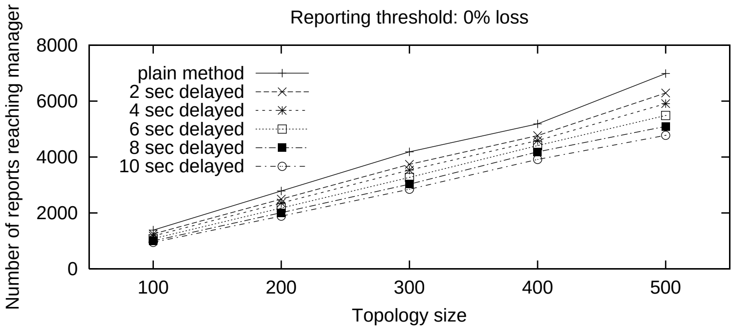

Fig. 4. Plain vs Delayed: Reports reaching manager during 100 sec simulation.

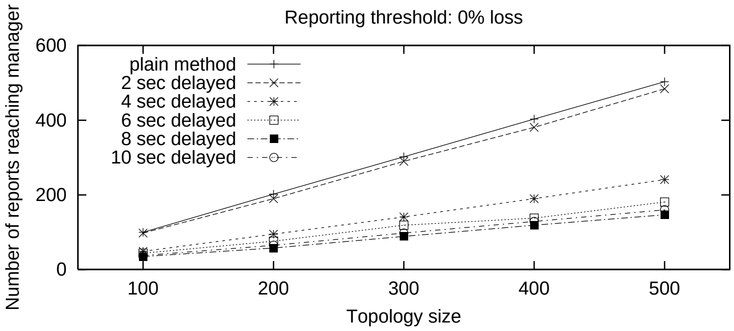

Fig. 5. Plain vs Delayed: Maximum reports reaching manager in 1 sec interval.

In the last set of simulations, we compare the plain report collection approach with the probabilistic report collection approach. As we mentioned above, in the probabilistic approach, the manager configures agents to send feedback reports based on some probability. In our simulations, we use reporting probabilities as 80%, 60%, 40% and 20%. Figure 6 compares the two approaches based on the number of feedback reports arriving at the manager site during a 100 second simulation. As the reporting probability decreases, the number of messages arriving the manager decrease and therefore the amount of feedback information loss increases. Similarly, Figure 7 shows that as the reporting probability decreases, the maximum number of reports reaching the manager site within a one second interval decreases. These results show that the probabilistic approach loses a significant number of feedback reports. Therefore, this method should only be used to get some approximate information about the reachability status of the network.

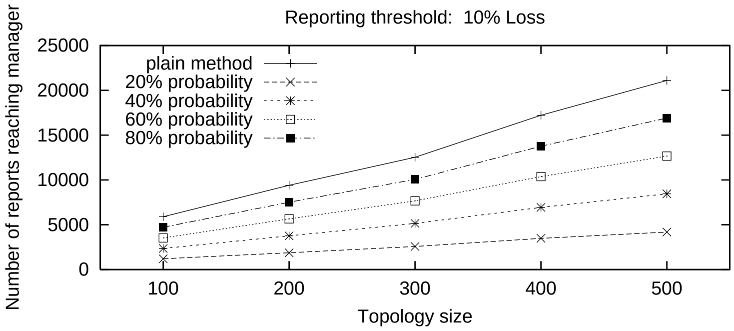

Fig. 6. Plain vs Probabilistic: Reports reaching manager during 100 sec simulation.

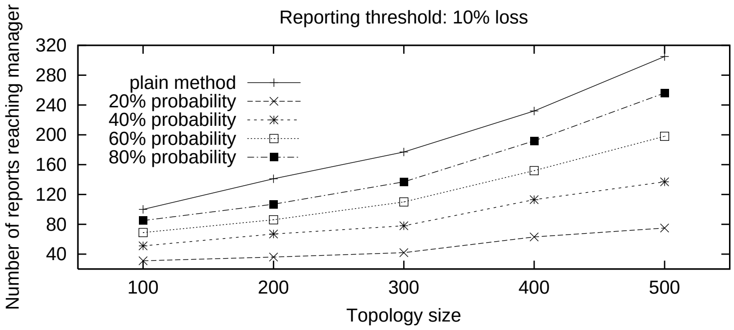

Fig. 7. Plain vs Probabilistics: Maximum reports reaching manager in 1 sec interval.

6 Discussion

The feedback report collection mechanism is the most important step in a reachability monitoring system. The complexity and the functional overhead of a monitoring system mostly depends on this component. Therefore, it is very important to choose the most appropriate feedback report collection technique. This decision usually depends on the needs of individual monitoring tasks. It may often be necessary for a reachability monitoring system to implement more than one feedback collection approach. It is then up to the user of this system to choose the correct alternative for a particular monitoring task. In the rest of this section, we provide some general suggestions about usage scenarios for each of the techniques that we discussed in this paper:

– *Plain reporting.* Plain feedback reporting should be used for test scenarios that have a small number of test receivers and the monitoring task requires collecting feedback information from all the agents.

– *Report suppression.* For monitoring tasks with a larger number of test receivers, report suppression approach provides more scalability. This approach should be used for monitoring scenarios where it is necessary to collect all the feedback information with better scalability than the plain reporting approach. As we mentioned previously, the report suppression method enables the manager to learn the existence and the approximate location of a threshold violation event but it does not provide information about which agents are affected by this event.
– *Delayed reporting.* Delayed reporting is most suitable for monitoring tasks where threshold violation events occur infrequently. In the delayed reporting approach, the manager receives feedback reports from all the test receivers. Furthermore, feedback reporting scalability can be adjusted by using different delay interval values. As the delay interval increases, the chance of report implosion at the manager site decreases. In this approach, test receivers may lose some feedback reports due to frequent threshold violations. For threshold violation events that occur in the backbone of the multicast tree (rather than the edge of the tree), feedback loss at a test receiver may not be very important. This is because another agent that has been affected by this threshold violation event can inform the manager about it. On the other hand, feedback losses occuring at the edge of the tree may not be recovered. Therefore, for monitoring tasks that are interested in learning threshold violation events in the core of the tree, this approach should work quite well.
– *Probabilistic reporting.* The probabilistic approach can be used for monitoring tasks that involve a large number of test receivers and the manager needs only approximate information about reachability. Furthermore, feedback implosion at the manager site can be controlled by using different reporting probabilities for the agents. In addition, agents can be assigned different reporting probabilities based on their strategic locations in the network.

As we have seen above, monitoring operations have potential scalability problems. These tasks may require configuring and managing a large number of monitoring agents. In addition, monitoring agents may be configured to send feedback reports based on some threshold violation and this often worsens the scalability problem inherent in reachability monitoring. In a related recent study, Al-Shaer and Tang proposed adding multicast as a communication mechanism in more traditional protocols like SNMP[3]. This improvement provides scalability in the one-to-many communication between a management station and a set of monitoring agents during the agent configuration step. The issue then becomes improving scalability of the many-to-one feedback collection mechanism. In our discussion, we use reachability monitoring tasks as example and evaluate a number of alternative approaches for many to one report collection. We believe that our results applies to other monitoring tasks having many to one feedback collection components.

7 Conclusions

In this paper, we have presented a general architecture for multicast reachability monitoring systems and provided a detailed discussion about the functional components and alternative implementations for these components. We have described different approaches to implementing the functionality in each step. The differences between these implementations are mainly based on the type and scope of different monitoring tasks. For the agent configuration and actual monitoring steps, the two important considerations are the active vs. passive nature of the monitoring task and the computational resources available at the agent platforms. On the other hand, the feedback report collection step is the most important step of a monitoring system as it affects the overall performance of the monitoring system. In the paper, we presented four different approaches with different scalability and information completeness characteristics for the feedback report collection step. In addition, we performed simulations to compare the performance of these different feedback report collection techniques in terms of their scalability characteristics. Based on the simulation results, we provided suggestions on when to use which feedback reporting technique.

References

1. J. Case, K. McCloghrie, M. Rose, and S. Waldbusser, "Protocol operations for version 2 of the simple network management protocol (SNMPv2)." Internet Engineering Task Force (IETF), RFC 1905, January 1996.
2. J. Case, R. Mundy, D. Partain, and B. Stewart, "Introduction to version 3 of the internet-standard network management framework." Internet Engineering Task Force (IETF), RFC 2570, April 1999.
3. E. Al-Shaer and Y. Tang, "Toward integrating IP multicasting in internet network management protocols," *Computer Communications–Integrating Multicast into the Internet*, 2000.
4. K. Almeroth, L. Wei, and D. Farinacci, "Multicast reachability monitor (MRM)." Internet Engineering Task Force (IETF), draft-ietf-mboned-mrm-*.txt, October 1999.
5. V. Paxson, J. Mahdavi, A. Adams, and M. Mathis, "An architecture for large-scale internet measurement," *IEEE Communications*, August 1998.
6. H. Schulzrinne, S. Casner, R. Frederick, and J. V., "RTP: A transport protocol for real-time applications." Internet Engineering Task Force (IETF), RFC 1889, January 1996.
7. A. Adams, R. Bu, R. Caceres, N. Duffield, T. Friedman, J. Horowitz, F. Lo Presti, S. Moon, V. Paxson, and D. Towsley, "The use of end-to-end multicast measurements for characterizing internal network behavior," *IEEE Communications*, May 2000.
8. 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.
9. J. Walz and B. Levine, "A hierarchical multicast monitoring scheme," in *International Workshop on Networked Group Communication (NGC)*, (Palo Alto, California, USA), November 2000.
10. M. H. Ammar, "Probabilistic multicast: Generalizing the multicast paradigm to improve scalability," in *IEEE Infocom*, (Toronto, CANADA), June 1994.
11. E. Zegura, C. K., and D. M., "Modeling internet topology," tech. rep., College of Computing, Georgia Institute of Technology, 1999.
12. M. Yajnik, J. Kurose, and D. Towsley, "Packet loss correlation in the MBone multicast network," in *IEEE Global Internet Conference*, (London, ENGLAND), November 1996.