A Distributed Approach for Monitoring Multicast Service Availability

Kamil Sarac
Dept of Computer Science
University of Texas at Dallas
Richardson, TX 75083
ksarac@utdallas.edu

Kevin C. Almeroth
Dept of Computer Science
University of California
Santa Barbara, CA 93106
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. One of the most important management tasks for multicast is to verify the availability of the service to its users. This task is usually referred to as reachability *monitoring. Reachability monitoring requires a number of monitoring stations to work together to collect this information in a distributed manner in the inter-domain scale.*

In this paper we present 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. Using simulations, we compare a number of alternative approaches for feedback collection and make suggestions on when to use each. We believe our work provides 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.

Multicast data 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 [1]. 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. We identify the functional components/steps of this architecture as (1) agent configuration, (2) actual monitoring, and (3) feedback collection. In our discussion, we provide a number of alternative approaches to implementing each of these steps. Then we focus on the feedback collection component. To a large extent, this step determines the complexity and the overhead of a reachability monitoring system. We use simulations to compare a number of alternative approaches for feedback collection and make suggestions on when to use each approach. This paper extends our previous work [2] by including new types of feedback collection mechanisms and improving our simulations to better understand the advantages and disadvantages of alternative feedback collection mechanisms. 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 and gives a detailed discussion on alternative approaches for implementing the functional steps of monitoring systems. Section 4 presents our simulation-based evaluations of different feedback collection approaches and the paper concludes in Section 5.

2 Motivation

From a 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. 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, management 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 Distributed Multicast Reachability Monitoring

In this section we present a general architecture for distributed (inter-domain scale) multicast reachability monitoring systems. In this architecture, we include 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 [3]. 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. Below, we discuss these steps in more detail.

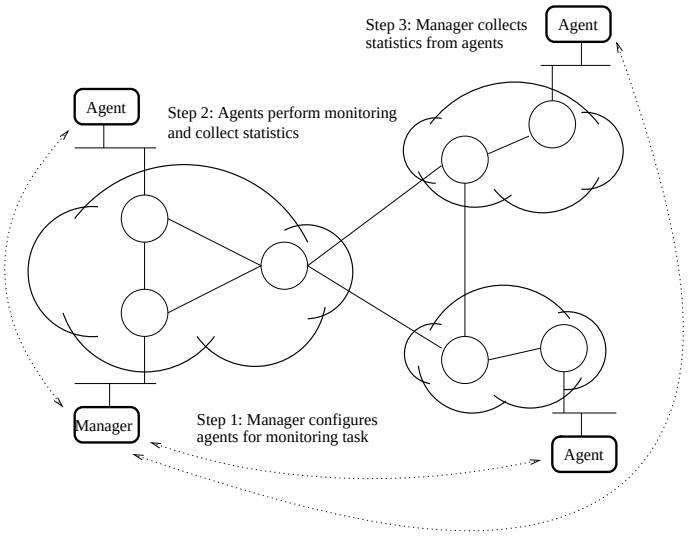


Figure 1. A general case for inter-domain reachability monitoring.

3.1 Step I: Agent Configuration

In this step, the manager configures the agents for a particular monitoring task by sending the 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. 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-based (IPsec) 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 [4]. It is a common piece in many network management systems. Most Network Operation Center (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 [5]. Second, there are research efforts looking to improve SNMP's distributed management capabilities—primarily by adding scalability [6]. We believe that with these modifications, SNMP can provide solid vehicle for control/configuration communication for distributed 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 [7]. 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 [8]. In NIMI, monitoring platforms are protected by a password mechanism and (AH-based) authenticated message exchanges. We believe that NIMI infrastructure

is one very good example platform to conduct distributed multicast reachability monitoring tests discussed in this paper.

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.

3.2 Step II: Reachability Monitoring

In this step, agents perform the actual monitoring and collect reachability data. 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. 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 [9]. In a related joint study, we proposed a number of extensions to the Real-time Transport Protocol (RTP) reporting mechanism [10] that could provide a standard model to collect more detailed information about multicast data reception quality.

3.3 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 send them to the manager at a later time. In this approach, in order to reduce the possibility of having a feedback implosion problem at the manager site, agent feedback is collected asynchronously.

Real-time reporting, on the other hand, can be divided into two types: alarm-based reporting and polling-based reporting. In alarm-based reporting, agents create and send feedback reports based on detecting a threshold violation event (e.g. observed packet loss exceeds a given threshold value). In polling-based reporting, agents create and send a feedback report based on an explicit request from the manager. Contrary to the off-line reporting case, real-time feedback collection approaches suffer from potential implosion problems as discussed below.

3.3.1 Alarm-Based Feedback Collection

In distributed reachability monitoring, where a large number of agents are configured to send alarm-based 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 [11] (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. 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 [12].

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, the Real-time Transport Protocol (RTP) [13] 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. 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 [14].

In the above list, we briefly discussed alternative approaches for alarm-based 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.

3.3.2 Polling-based Feedback Collection

In the polling-based feedback collection approach, in order to learn current reachability information, the manager sends an explicit request message to the agents and expects to receive feedback information from them. Depending on the size of the agent population, the communication from the manager to the agents can use both unicast and/or multicast. Note that, even though a multicast-based control communication approach provides more scalability and less overhead in the network, potential reachability problems between the manager and the agent sites can limit its usability for this purpose.

Similar to the above case, in a monitoring task where a large number of agents are requested to send feedback information matching a given response criteria, the feedback reports originating from these agents can create scalability problems both in the network and at the manager site. An example polling request may ask all the agents currently experiencing more than 10% packet loss to send a feedback report back to the manager. When sending such a request message, the manager may not also know the number of agents matching the given response criteria.

The feedback collection approaches that we presented in the previous section can be used in this context with similar performance behavior. One additional overhead would be the network overhead caused by sending explicit feedback request messages by the manager. Depending on the size of the agent population, request messages can be sent via unicast or multicast. In both cases, the incurred overhead would be similar for the different approaches.

In addition to the above approaches, we can use a probabilistic multi-round approach for polling-based feedback collection. Below, we present this approach. For comparison reasons, we include a discussion on the use of the delayed approach for polling-based feedback collection.

1. **Probabilistic multi-round approach.** In this approach, the manager uses multiple rounds to collect feedback information. In the first round, the manager uses an initial guess about the total number of feedback messages to be collected and then computes a reporting probability and asks the agents to send their feedback with this probability. On receiving responses from a number of agents, it then improves its guess on the number of feedback messages to be collected and adjusts its reporting probability and sends a new request with this new probability value. This way, in each round the manager collects a manageable number of feedback messages and also improves its guess on the total number of feedback messages to be collected. Compared to the previous approach, this approach has better protection against feedback implosion problems but may incur additional traffic overhead in the network (in the form of sending new feedback requests in each round).

2. **Delayed approach.** In this approach, the manager sends a polling request to the agents. In this request, the manager indicates the maximum delay interval as well as the feedback reporting threshold. Each agent, upon receiving this feedback request, first examines the reporting condition and decides if it should send a feedback message or not. Agents that decide to sends their feedback then choose random delay values out of the given maximum delay interval and send their feedback at the end of this delay period. The main goal in using a random delay period is to disperse the agent feedback and reduce the possibility of having a report implosion problem at the manager site. However, this requires that the maximum delay interval value should be selected carefully. In other words, the manager should consider the desired number of simultaneous feedback messages to be received and the expected number of messages to be generated by the agents. But, in general, the number of feedback messages depends on the observations made at the agent sites and this information may not be known by the manager. A conservative alternative is for the manager to use the total number of agents as part of the delay interval computation.

The main difficulty with this approach is the fact that the manager may not know the number of feedback messages to be received and therefore the given maximum delay interval calculation may not be accurate all the time. An overestimate of the value may result in unnecessary delays in collecting the feedback and an underestimation may result in implosion.

Above, we briefly discussed the scalability characteristics of the alternative approaches. In addition to scalability, these approaches have different behavior when packets (especially request packets) are lost. In the case of the delayed approach (and all the other approaches presented in the previous section), if the request packet is lost before reaching a fraction of the agents, this will cause information loss for the manager. On the other hand, due to the multiple round nature of the probabilistic multi-round approach, the manager will have a better chance of collecting the missing information in additional rounds.

4 Evaluations

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 show trade offs of different feedback reporting mechanisms. For this, we compare feedback reporting approaches based on a number of metrics. These include: (1) traffic overhead in the network, and (2) message overhead at the manager site. In addition, we discuss relative performance of alternative approaches with respect to information granularity/completeness and timeliness. Furthermore, for the polling-based approach, we compare the alternative approaches based on their reaction to packet losses, specifically request packet losses. Note that alarm-based reporting does not use explicit feedback requests. We expect that our evaluations will help us better understand the relative performance of alternative approaches. This will then help us in choosing a proper feedback collection approach based on the needs/characteristics of a given monitoring task/scenario.

4.1 Evaluation of Alarm-based Feedback Collection Approaches

In this section, we use plain feedback reporting as the base case and compare the performance of the other alternatives to that of plain reporting. For comparisons we use network topologies with different sizes and different threshold violation values for alarm reporting.

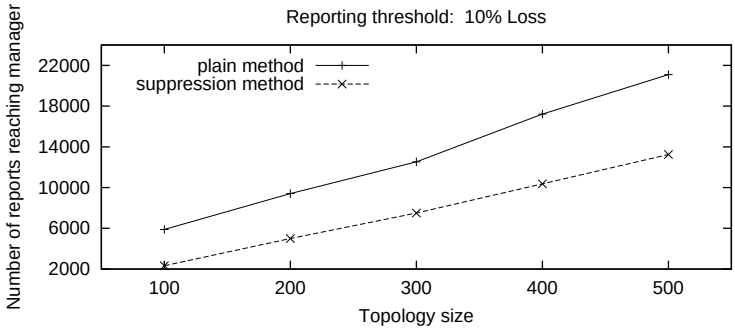
We generate random flat graphs for 100, 200, 300, 400 and 500 node topologies using the Georgia Tech Internet Topology Modeler (GT-ITM) [15]. In these topologies, we identify one of the nodes as the *test sender* and all the others as *test receivers*. Then, we assign loss probabilities for each link in the network. For loss assignment,

we use the guidelines presented in Yajnik et. al. [16]. 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 $(0 + \epsilon)\%$ 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.

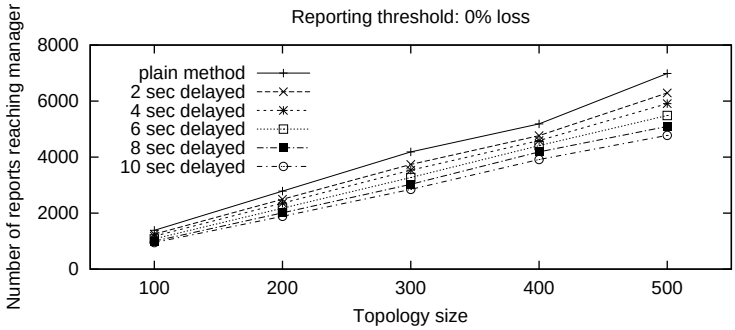
4.1.1 Traffic Overhead in the Network

The first set of simulations are run to compare the alternative approaches to the plain feedback collection approach based on their network overheads. Figure 2-a shows the results for the suppression-based approach. In our simulations, 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. 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 the report. Figure 2-a 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 savings of the suppression-based feedback collection technique. This savings is gained at the expense of having suppression functionality at every monitoring agent.

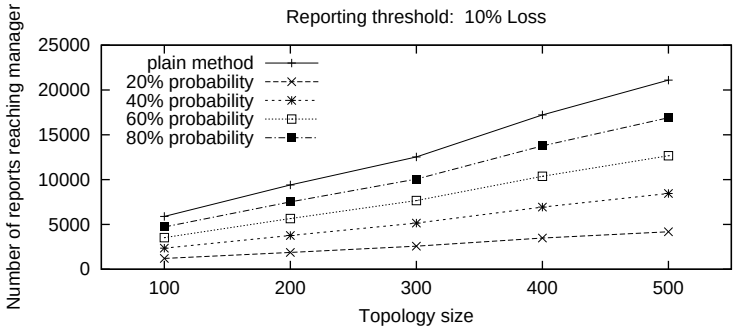
In the next set of simulations, we compare the plain report collection approach with the delayed feedback collection approach. In the simulations, we use 2, 4, 6, 8 and 10 second delay intervals. The actual delay values assigned to agents are randomly selected within these intervals. Figure 2-b shows 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 when their delay expires. But this introduces extra overhead into the system as agents need to keep track of all pending reports.



(a) Plain vs Suppression.



(b) Plain vs Delayed.



(c) Plain vs Probabilistic.

Figure 2. Network overhead: Message overhead during 100 sec simulation.

In the last set of simulations, we compare the plain report collection approach with the probabilistic report collection approach. In the simulations, we use reporting probabilities as 80%, 60%, 40% and 20%. Figure 2-c compares the two approaches based on the number of feedback reports propagating in the network toward the manager site during a 100 second simulation. As the reporting probability decreases, the number of messages (the network overhead) decrease but the amount of feedback information lost increases.

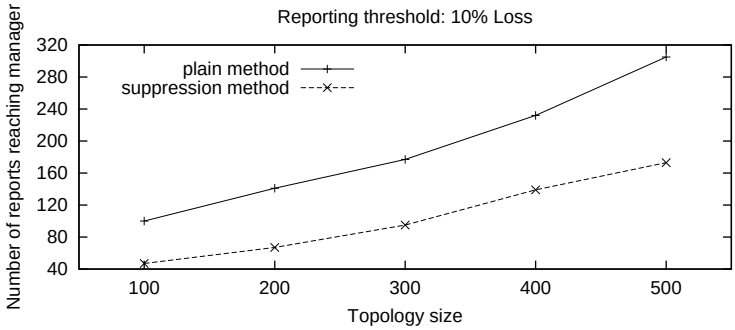
4.1.2 Messaging (Implosion) Overhead at the Manager Site

In this section, we present our simulation results to compare the alternative approaches to the plain feedback collection approach based on their message overhead at the manager site, i.e. feedback implosion at the manager site.

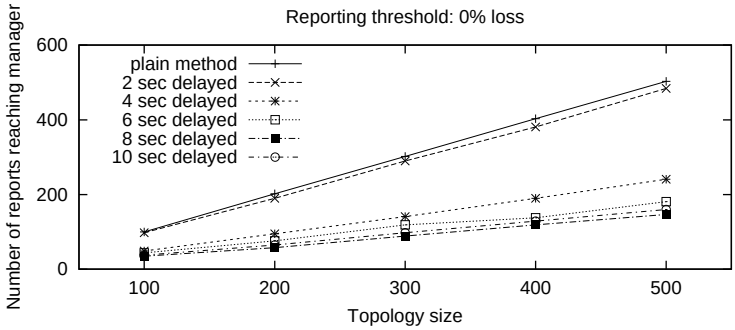
Figure 3-a compares the suppression-based feedback collection approach to the plain feedback collection approach based on the maximum number of feedback reports reaching the manager site within a one second interval. These results present the peak number of reports that the manager must process in one second. These results indicate that the suppression-based feedback collection approach provides better scalability for feedback report collection without introducing any loss of feedback information.

The next set of simulations present the level of feedback implosion incurred by the delayed feedback collection approach. Figure 3-b shows the maximum number of feedback reports reaching the manager site in a one second interval. According to this figure, the messaging overhead at the manager site decreases as the delay period increases. These results indicate that in the case of infrequent losses, the delayed feedback reporting approach provides more feedback scalability at the manager site. On the other hand, depending on the frequency of threshold violations, it may cause feedback information loss.

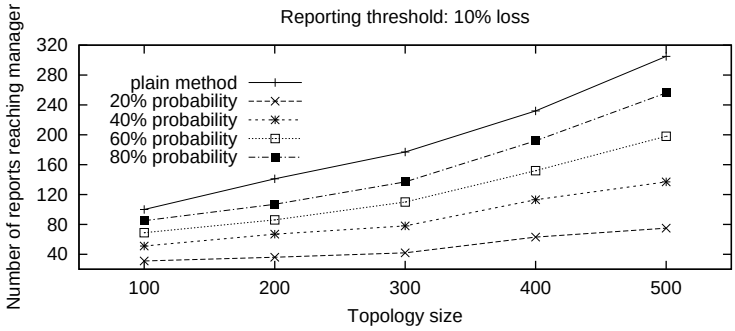
One interesting result in Figure 3-b is that the performance of the “plain method” and the ”2 sec delayed method” look very similar. This is mainly due to selecting a short delay interval. That is, in the simulations we use a one second interval to compute the loss ratios. In addition, for the “2 sec delayed method”, once a threshold violation is detected, the affected test agents set their timers for a feedback report. Figure 4 shows an example situation. According to this figure, X many agents detect a loss-based threshold violation event at time 1 and set their timers. Approximately $\frac{X}{2}$ many agents send their reports in the time 1 to 2 interval and the other half in the time 2 to 3 time interval. At this point assume that these agents detect another threshold violation in the time 1 to 2 interval. In this situation, they will again schedule their timers to send reports back to the manager. However, when we look at the time interval 2 to 3, we see that a total of X agents send their responses back to the manager. As a result, in the case



(a) Plain vs Suppression.



(b) Plain vs Delayed.



(c) Plain vs Probabilistic.

Figure 3. Message overhead: Max. reports reaching manager in 1 sec interval.

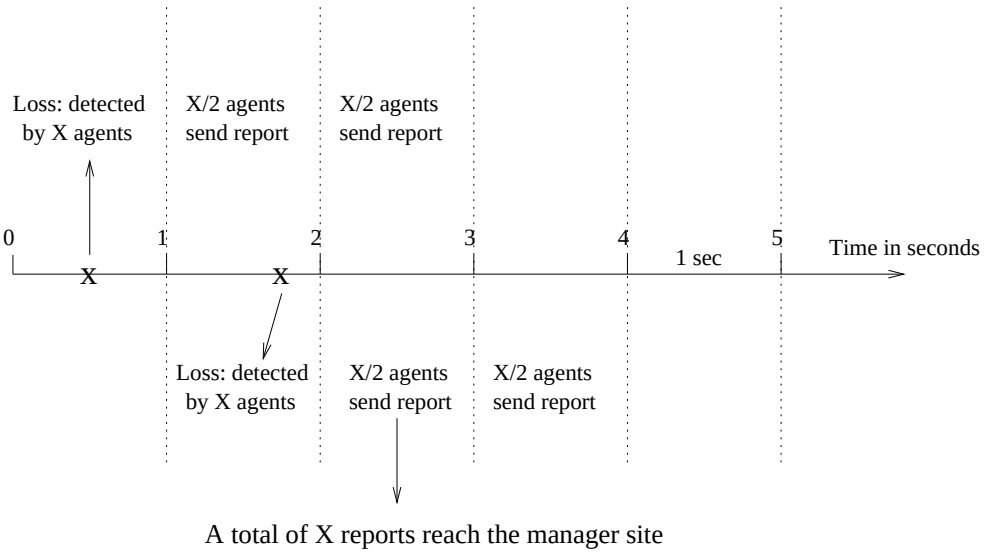


Figure 4. Effect of short delay intervals on peak number of response.

of frequent threshold violations, choosing a short delay interval does not really help us to reduce the peak number of responses going back to the manager site.

The last set of simulations compare the plain report collection approach with the probabilistic report collection approach. Figure 3-c shows that as the reporting probability decreases, the maximum number of reports reaching the manager site within a one second interval decreases. By reducing the feedback probability, the implosion level at the manager site can be controlled. However, this may cause feedback information loss for the manager.

4.1.3 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 occurring 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.

4.2 Evaluation of Polling-based Feedback Collection Approaches

In this section, we compare the performance of the probabilistic multi-round approach and the delayed approach mechanisms based on different evaluation metrics. The evaluation metrics include the traffic overhead in the network, message overhead at the manager site, and the reaction to request packet losses in the network (i.e. information completeness on the face of packet losses in the network).

4.2.1 Traffic Overhead in the Network

In the polling-based approach, traffic overhead in the network includes request messages from the manager and the responses from the agents. The traffic overhead due to agent responses depends on the reporting criteria set

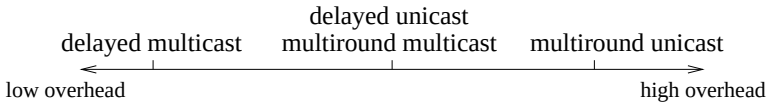


Figure 5. Network overhead: Messages exchanged in the network.

by the manager and this overhead equals to the number of agents that satisfy the reporting criteria. Therefore, the overhead due to responses is the same for both the multi-round approach and the delayed feedback approach. On the other hand, the traffic overhead due to request packets may differ between the two approaches. In the multi-round approach, it takes multiple round and in the delayed approach, it takes only one round to collect the feedback information. Therefore, the traffic overhead due to request packets is usually smaller in the delayed approach than in the multi-round approach. In addition, the request overhead depends on the communication service used. In general multicast-based requests incur less overhead than unicast-based requests. Figure 5 presents the relative performance of the two alternative approaches.

4.2.2 Messaging (Implosion) Overhead at the Manager Site

In this section, we compare the two approaches based on their message overhead (feedback implosion level) at the manager site. As we mentioned previously, the messaging overhead depends on the number of agents, R, that match the reporting criteria set by the manager. In general, R is not known by the manager in advance. This fact has different implications on the performance of feedback collection approaches.

In the delayed approach, the manager uses an initial guess $\bar{R}$ for the number of responding agents. Depending on the accuracy of $\bar{R}$, the message overhead at the manager site may change significantly. In the multi-round case, on the other hand, the manager uses an initial guess, $\bar{R}$, to compute the response probability for the agents. Based on this probability, it receives a number of responses in the first round. Then, by using the number of responses received, it modifies its guess to better approximate the R value. This way, it achieves a better control on the implosion level. Figure 6 presents the manager's estimation for the number of reporting agents matching a given reporting criteria.

In order to compare the two approaches based on their message overhead, we run a second set of simulations. In these simulations, we use a monitoring scenario with 1,000 agents. The implosion threshold value at the manager site is set to K=50. Then, we randomly choose a number of nodes (500 or 750) as reporting agents. These are the agents which correspond to the monitoring agents that match the reporting criteria in a real world monitoring scenario. Since sending out a feedback report is a local decision at an agent site, the number of such agents may not

K $\leftarrow$ desired num. responses
$X_i \leftarrow$ num. returned responses in a round
$\bar{R}_i \leftarrow$ manager's estimation of num. reporting agents
$P_i \leftarrow$ reporting probability

$$\bar{R}_i \leftarrow \frac{X_{i-1}}{P_{i-1}} + \sum_{j=0}^{i-2} x_j$$
$$P_i \leftarrow \frac{K}{R_i - \sum_{j=0}^{i-1} x_j}$$

Figure 6. Manager's estimation of num. agents in multi-round approach.

be known by the manager. In this case, the manager starts with an initial guess of the number of such agents. And, the initial guess of the manager may result in underestimation, correct estimation, or overestimation of the incoming feedback reports. Tables 1, 2, and 3 present simulation results for these cases respectively. The first three columns presents the maximum, the average and the median number of feedback messages reaching the manager site in a round. The next two columns present the standard deviation of responses and the number of rounds respectively. Then, the last two columns represent the request loss probability at the agent sites and the feedback messages loss at the manager site respectively. Feedback message loss is mainly due to request packet loss at the agent sites. That is, if an agent does not receive a request packet, it will not send a response message. Even if this agent satisfies the response criteria, this information will not be delivered to the manager which we call as response packet loss.

According to these tables, in the multi-round approach, the average and the median number of feedback reports reaching the manager site in a round is close to the given threshold value (K=50) in all three tables. On the other hand, in the delayed-based approach, these numbers cause unnecessary delays (more than necessary number of rounds) or feedback implosion (significantly more than K feedback reports) in the overestimation (shown in Table 1) and underestimation (shown in Table 3) cases respectively. If the initial guess of the manager in the delayed-based approach is not good enough, it will affect the overall feedback collection operation in one of these two ways.

4.2.3 Reaction to Packet Loss

The final evaluation metric is the reaction to request packet losses in the network. As we mentioned previously, due to the single round nature of the delay-based approach, when a request message is lost before reaching a group of the agents, it will potentially cause information loss for the manager. On the other hand, due to the multi-round nature of the probabilistic multi-round approach, a request packet loss in a round is usually compensated for in additional rounds and therefore agents have more chances of receiving request packets. Tables 1, 2, and 3 present the number of lost packets for each alternative approach with increasing packet loss probabilities at the agent sites (shown in the

#Reporting Agents (RA): 250; Mgr's guess of RA: 500; K: 50

Multi-round Approach						
Max	Avg	Median	StDev	#Round	Loss Prob	#Response Pkt Lost
66	41.66	47	18.81	6	0	0
72	41.5	37	21.81	6	0.05	1
67	49	54	17.3	5	0.1	5
68	46	41	16.39	5	0.2	20
Delayed-based Approach						
Max	Avg	Median	StDev	#Round	Loss Prob	#Response Pkt Lost
31	25	24.5	3.28	10	0	0
31	23.8	25	4.72	10	0.05	12
27	21.8	22.5	3.42	10	0.1	32
28	20.7	21.5	4.1	10	0.2	43

Table 1. Message overhead: Over-estimation of num. of reporting agents.

last two columns in each table). According to the tables, the response loss in the delayed approach is closely related to loss probability. In the multi-round approach the number of lost packets is more related to the number of rounds. As the number of rounds increases, the amount of response loss decreases.

5 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 several 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 techniques in terms of their scalability characteristics. In this work, we use reachability monitoring as an example task to evaluate a number of alternative approaches for many to one report collection. We believe that our results apply to other monitoring tasks having many-to-one feedback collection components.

#Reporting Agents (RA): 500; Mgr's guess of RA: 500; K: 50

Multi-round Approach						
Max	Avg	Median	StDev	#Round	Loss Prob	#Response Pkt Lost
79	50	46	12.51	10	0	0
68	49.8	49.5	9.41	10	0.05	2
59	49.5	49.5	5.97	10	0.1	3
68	49	50	9.77	10	0.2	10
Delayed-based Approach						
Max	Avg	Median	StDev	#Round	Loss Prob	#Response Pkt Lost
57	50	51.5	5	10	0	0
56	47	46.5	5.63	10	0.05	30
63	44.9	46	2.96	10	0.1	51
49	40.3	39.5	4.6	10	0.2	97

Table 2. Message overhead: Perfect estimation of num. of reporting agents.

#Reporting Agents (RA): 750; Mgr's guess of RA: 500; K: 50

Multi-round Approach						
Max	Avg	Median	StDev	#Round	Loss Prob	#Response Pkt Lost
83	50	47	14.52	15	0	0
76	50	47	13.47	15	0.05	0
76	49.8	49	16.19	15	0.1	3
81	52.85	50.5	14.29	14	0.2	10
Delayed-based Approach						
Max	Avg	Median	StDev	#Round	Loss Prob	#Response Pkt Lost
86	74.9	74.5	7.7	10	0	0
86	74.9	74.5	7.7	10	0.05	0
83	67.9	70.5	9.61	10	0.1	71
71	58.9	57.5	8.72	10	0.2	161

Table 3. Message overhead: Under-estimation of num. of reporting agents.

References

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

[2] K. Sarac and K. Almeroth, "Providing scalable many-to-one feedback in multicast reachability monitoring systems," in *4th IFIP/IEEE International Conference on Management of Multimedia Networks and Systems (MMNS)*, (Chicago, IL, USA), October/November 2001.

[3] P. Sharma, E. Perry, and R. Malpani, "Ip multicast operational network management: design, challenges, and experiences," *IEEE Network*, vol. 17, pp. 49–55, March/April 2003.

[4] 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.

[5] 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.

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

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

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

[9] 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.

[10] 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.

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

[12] K. Sarac and K. Almeroth, "Tracetree: A scalable mechanism to discover multicast tree topologies in the internet," *IEEE/ACM Transactions on Networking*. (To appear).

[13] 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.

[14] M. H. Ammar, "Probabilistic multicast: Generalizing the multicast paradigm to improve scalability," in *IEEE Infocom*, (Toronto, CANADA), June 1994.

[15] E. Zegura, C. K., and D. M., "Modeling internet topology," tech. rep., College of Computing, Georgia Institute of Technology, 1999.

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