Multicast Feedback Suppression Using Representatives

Dante DeLucia
Computer Science Department
University of Southern California
Los Angeles, CA 90089-0781
and
Hughes Research Laboratories
3011 Malibu Canyon Road
Malibu CA 90265
voice/fax: (805) 682-6834
email: dante@usc.edu

Katia Obraczka
University of Southern California
Information Sciences Institute
4676 Admiralty Way Suite 1001
Marina Del Rey, CA 90292
voice: (310)822-2301 fax: (310)823-6714
email: katia@isi.edu

August 16, 1996

Abstract

For a reliable, flow-controlled multicast transport protocol to scale, it must avoid the feedback implosion problem [5], particularly if the protocol targets arbitrarily large multicast groups communicating over lossy networks.

Most existing feedback control mechanisms based on probabilistic suppression address the feedback implosion problem by suppressing feedback using timers based on round-trip time (RTT) information. This approach requires that all receivers compute RTT to the data source.

We present an algorithm whose major benefit derives from the fact that it does not need to compute RTT from receivers to the source, and does not require knowledge of group membership or network topology. We use a small set of *representative* receivers and probabilistic suppression to limit feedback.

We believe that our approach will perform well in real networks. Simulations using randomly-generated network topologies of varying sizes with pessimistic network loss rates show that representatives considerably reduce the amount of feedback compared to a purely suppression-based scheme. For various multicast group sizes, a few representatives can keep the amount of feedback low while not degrading feedback timeliness.

Keywords: Multicast, feedback, suppression, representatives

1 Introduction

While many existing multicast transport protocols such as SRM [7], and RTP [17], target delay sensitive, real-time applications, little has been done to address multipoint bulk data transfer services, such as multicast FTP.

The feedback suppression algorithm we present is part of our efforts to build a scalable flow controlled multicast transport mechanism suitable for bulk data transfer applications. Our solution addresses the problem of feedback implosion[5], while providing the frequent and timely feedback required by a flow control algorithm. Our solution does not require knowledge of the multicast group membership or network support.

Several existing reliable multicast transport protocols, such as SRM [7], use RTT-based probabilistic suppression to avoid feedback implosion at the source. In RTT-based suppression, each receiver measures RTT to the source and sets a feedback timer based on its distance from the source. Receivers nearer the source will be the first to respond to a packet loss and will tend to suppress NACKs from receivers farther down the multicast tree. This approach is intended for conferencing applications in which many-to-many communication takes place. In this environment, computing RTTs to the source requires little additional overhead.

In this paper, we present a solution to the feedback implosion problem that does not rely on RTT estimation and hence has greater potential for scalability. Although our mechanism is not tied to any particular application, it is aimed at applications with a single data source and multiple receivers. All communication is via multicast.

We use a small set of group members, or *representatives*, to represent the congested multicast subtrees. In addition to providing fast feedback to the source, representatives suppress feedback from the other group members. Through simulations, we evaluate our feedback suppression mechanism in terms of the amount and timeliness of the feedback generated by the multicast group.

For multicast groups of varying sizes communicating over randomly-generated network topologies subject to pessimistic loss rates, our simulation results show that representatives can greatly reduce the amount of feedback when compared to purely suppressive algorithms, yet still provide timely feedback to the source. Representatives also make the algorithm less sensitive to suppression timer settings, which allows the algorithm to perform well without having to rely on RTT estimation between each receiver and the source. Not performing RTT estimates cuts down the overhead of our feedback control mechanism considerably. Our simulations also show that representative set size scales well with multicast group size.

2 Related Work

2.1 Feedback Control

Traditional reliable unicast transport protocols, such as TCP, use ACKs to recover from packet loss. This approach to achieving reliability is often referred to as *sender-initiated*, since it is the responsibility of the sender to detect packet losses. In a multicast environment, as group sizes increase, the sender-initiated scheme may cause ACK implosion since each delivered packet triggers an acknowledgment from every receiver in the group.

Alternatively, in the *receiver-initiated* approach to reliability, receivers detect packet losses and request its retransmission by generating a NACK. Placing the responsibility of recovering from packet losses on the receiver helps alleviate the ACK implosion problem. The performance comparison study presented in [15] confirms that receiver-initiated multicast transport protocols have better scalability propoerties than their sender-initiated counterparts. For this reason, most reliable multipoint transport protocols are either pure receiver-initiated or use a hybrid approach by combining receiver- and sender-initiated reliability.

However, receiver-based protocols also suffer from the feedback implosion problem, especially when losses occur higher up in the multicast tree in larger groups over lossy networks. In this section we focus on proposed solutions to the feedback implosion problem in the context of reliable multicast transport protocols.

In [8], solutions to the feedback implosion problem are classified as *structure-based* or *timer-based*. Structure-based approaches, such as the Log-Based Protocol [10], rely on a designated site (either a dedicated server in the case of the Log-Based Protocol [10] or a pre-assigned group member) to process and filter feedback information.

Timer-based solutions rely on probabilistic feedback suppression to avoid implosion at the source. Receivers in the SRM protocol [7], which was designed to support the WB distributed whiteboard application, delay their retransmission requests for a random interval, uniformly distributed between the current time and the one-way trip time to the source. The goal is that group members closer to the source send their feedback sooner suppressing feedback from farther away members. A site uses periodic *session messages* to measure its distance (based on the resulting RTT) to the other group members.

The Deterministic Timeouts for Reliable Multicast (DTRM) [8] algorithm also uses RTT between receivers and the sender to compute the receivers' suppression timeouts.

The feedback control mechanism proposed in [13] does not fall into either the structure-based or the timer-based categories. In this approach to feedback control, which is used by the IVS videoconferencing tool [18] and is layered atop of RTP [17], video sources use probabilistic polling to select a set of receivers that should provide feedback.

2.2 Window-Based versus Rate-Based Flow Control

We have considered two approaches to flow control. The first is the traditional window-based flow control scheme used by TCP. The second approach is the rate-based scheme such as that used in NETBLT [4].

Window-Based Flow Control

A more traditional approach to flow control is allowing the sender to transmit a certain number of packets, or a window, at a time. The sender advances its window each time it receives an acknowledgment for an outstanding packet. The problem of using a window-based flow control scheme in a multicast environment is deciding how to adjust the transmission window.

Because of the ACK implosion problem, multicast transport protocols use negative acknowledgment (NACKs) to signal packet loss. Should the window be adjusted in the absence of a NACK? How long do we wait for NACKs? We could measure the number of NACKs coming back, but since we do not know the size of the group, counting NACKs is not very useful.

Another conceptual problem is that a window-based flow control mechanism is closely tied to reliability. There has been a number of arguments in the literature for decoupling flow control from reliability. For these reasons, we decided to use a rate-based flow control scheme.

Rate-Based Flow Control

In rate-based flow control, the transmission rate can be set independently of the reliability mechanism. The source sends packets according to the current transmission rate. Since our goal is to avoid packet loss, the receivers should determine how much queuing is taking place in the network. If there is too much data queued in the network, receivers need to notify the source to slow down.

Since packet losses may still occur, the flow control mechanism should respond by multiplicatively decreasing the transmission rate when NACKs are received at the sender.

2.3 Congestion Control

In [14], Nagle shows that congestion can occur even in a datagram network with infinite storage. Jain et al. [16] proposes a congestion avoidance scheme, where routers signal congestion by setting the *congestion avoidance bit* in the packet's network-layer header. Depending on the overall feedback received, sources decide whether to increase or increase the current window size. Jain's network-layer congestion avoidance scheme requires a new bit in the packet headers as well as routers being able to set this bit.

In [12], Jacobson describes his slow-start flow control algorithm for TCP, which gradually opens the TCP transmission window as the source receives acknowledgments from the receivers. Slow-start uses data loss as sign of congestion and shuts the window down to 1 packet after a packet is lost. Both the Tahoe and Reno distribution of BSD UNIX [11] incorporate Jacobson's slow-start algorithm.

In multicast communication, the probability of loosing packets grows as a function of the group size, and the cost of a packet loss is much higher than in a point-to-point exchange. Because reliable delivery is critical for data distribution applications, our goal is to avoid packet drops.

Another variant of TCP called TCP Vegas [3] implements a sender-side congestion avoidance algorithm. In [1], Danzig et al. confirms that Vegas' congestion avoidance scheme yields higher throughput and keeps less data in the network than Reno. By computing the difference between *best* and *current* round-trip times (RTT), a Vegas sender measures the amount of data queued in the network and adjusts its transmission window accordingly.

This congestion avoidance approach in a multicast environment does not scale well. Senders need ACKs to measure RTTs, which for large groups, may lead to ack implosion. There is also the unknown membership problem. If group members' clocks were synchronize, it would be possible to measure one-way trip times, and have receivers compute queuing themselves. However, clock synchronization requires an additional protocol. In Section 4, we describe some techniques to address the scalability problem.

2.4 Reliable Multicast Transport Protocols

Traditional reliable unicast transport protocols, such as TCP, use positive acknowledgments (ACKs) to recover from packet loss. This approach to achieving reliability is often referred to as *sender-initiated*, since it is the responsibility of the sender to detect packet losses. In a multicast environment, as group sizes increase, the sender-initiated scheme may cause *acknowledgment implosion* since each delivered packet triggers an acknowledgment from every receiver in the group.

Alternatively, in the receiver-initiated approach to reliability, receivers detect packet losses and request its retransmission by generating a negative acknowledgment. Placing the responsibility of recovering from packet losses on the receiver alleviates the acknowledgment implosion problem. The performance comparison study presented in [15] confirms that receiver-initiated multicast transport protocols deliver better performance than their sender-initiated counterparts.

Most reliable multipoint transport protocols are either pure receiver-initiated or use a hybrid approach by combining receiver- and sender-initiated reliability. Below, we overview some of these protocols.

3 The Model

Our feedback suppression mechanism focuses on the following application-level requirements and lower-layer services.

- IP Multicast:

  Senders transmit data packets using internet multicast. The current multicast routing model which has been in use on hundreds of routing domains that form the Internet's MBONE[1] is based on DVMRP, a distance-vector multicast routing protocol. DVMRP builds source-rooted shortest-path distribution trees, where all leaf routers are attached to group members. IP multicast uses the Internet Group Management Protocol (IGMP)[2] to manage group membership. Hosts send an IGMP join message to the multicast group they want to join. Multicast-capable routers use IGMP messages to propagate membership information among themselves and to poll directly attached hosts for updated membership information.

  Although we are assuming the current Internet multicast model, our protocol will work with any of the alternate multicast models that have been proposed, such as CBT [2] and PIM [6].

- Unknown Group Membership:

  To support scalability, it is assumed the set of receivers is unknown. This is also a reflection of the semantics of IP multicast where there is no centralized group management.

- Unknown Network Topology:

  No knowledge of the underlying physical network topology is assumed. Routers are not relied on to provide feedback about the network conditions, or filter feedback requests.

- Static Data:

  Our focus is on applications that distribute static data as opposed to real-time data. A multicast file distribution service, where files can be of arbitrary size, is a typical target application.

  We assume that application semantics issues, such as data consistency, will be handled by the specific application.

- Per-source Flow Control:

  Although there can be multiple sources in a multicast group, flow control is on a per-source basis. In other words, we are not designing an aggregate flow control mechanism for a multicast group.

[1] http://www.research.att.com/mbone-faq.html
[2] http://www.cis.ohio-state.edu/htbin/rfc/rfc1112.html

- Application Semantics

  Applications are responsible for the protocol semantics. When moving from a one-to-one reliability protocol such as TCP to a one-to-many protocol that would be used in multicast applications, it becomes much more difficult to build a single transmission protocol that can handle all the possible semantics that might be required.

  Even though we are targeting bulk data transfer applications, the model is very general. It can be applied to any multicast application requiring prompt feedback.

4 Feedback Suppression Algorithm

Our multicast feedback mechanism is based on the assumption that in a large multicast group, a small set of bottleneck links will cause the majority of the congestion problems. We exploit this by finding a small set of group members to represent the congested multicast subtrees. These group *representatives* provide immediate feedback which can suppress feedback from other group members thus preventing feedback implosion at the source.

If a receiver never experiences any packet loss, or has its packet losses covered by a representative, it will never generate any messages.

The first challenge in selecting representatives is to choose them such that they represent the congested subtrees of the multicast tree. Ideally, each congested link would be represented by one representative in the affected subtree. The second challenge is to react to new congestion in a timely manner by choosing new representatives and discarding those that are no longer contributing to the feedback efforts.

The congestion avoidance protocol we envision relies on both positive (ACK) and negative (NACK) acknowledgments. ACKs are required to prevent congestion collapse, while NACKs are required to provide feedback in the case of congestion. If a source does not hear any feedback, it can assume that either there are no group members other than itself, or that there has been some sort of catastrophic network failure.

Receivers that have been selected as representatives provide immediate feedback to the source. Feedback from other receivers is scheduled over a random interval and is subject to suppression. We explain how suppression timers are set in Section 4.5.

At startup, there is no representative set and suppression timers are set very loosely.[3] As feedback comes in, the source builds the representative set. In the absence of NACKs, the receivers whose ACKs were received by the source will be selected as representatives. Since NACKs are an immediate indication

[3]In our simulation we initially set the suppression timers to 1500 milliseconds. Once the GRTT (see 4.5.1) is computed we use that as the basis to set timers.

of congestion, feedback suppression will give precedence to NACKs over ACKs. Receivers sending NACKs will take precedence over receivers sending ACKs for consideration as representatives. As network conditions change, new feedback is received by the source and the representative set is updated.

4.1 The Source

The source maintains the representative set and computes the group's current maximum round-trip time (GRTT) defined in Section 4.5.1. The source is responsible for distributing the current representative set and the GRTT to the group.

4.2 Receivers

Upon receiving a data packet, non-representative receivers schedule a NACK if a data packet N is received without having seen packet N-1[4]. Otherwise an ACK is scheduled. A scheduled response is held for a random period of time before being sent. If another response is received before the scheduled send time and the received response is defined to be as "good or better" than the response scheduled, the scheduled response is suppressed. In our definition of "good or better", NACKs suppress NACKs and ACKs, while ACKs can only suppress other ACKs.

4.3 Representatives

Receivers designated as representatives send feedback to the source immediately forgoing any suppression interval.

A receiver designates itself as a representative when it receives a representative set notification in which it is a member. A representative reverts to non-representative operation when it receives a representative set update in which it is not a member.

4.4 Representative Selection

At startup, any receiver that provides feedback is eligible for selection as a representative. After a full representative set has been obtained, only NACKs qualify a receiver for selection as a representative. To prevent a sudden change of the representative set, only one new representative may be selected in response to any one packet. When the representative set is full and a new representative is selected, an existing representative must be ejected from the current set. The best candidate for ejection is obviously the "worst" representative, but the problem then lies in what constitutes the "worst". To keep things simple, the

[4]Since this is not a reliability mechanism, we only NACK the N-1 packet. We are not concerned with any other previous lost packets.

representative that has not sent a NACK in the longest time is selected. This criteria is based on the assumption that a representative that has not sent a NACK is having the fewest number of congestion problems.

4.5 Timers

Since we cannot compute RTT between the receivers and the source, we cannot use the "precise" RTT to set up the suppression timers as done in SRM [7]. So we are forced to resort to cruder measures. We can partially compensate for using loose timers by using representatives to provide fast feedback. Since representatives provide immediate feedback on behalf of the subtrees they cover, we need "backup" timers for those losses not covered by representatives.

We break our suppression times into two components. The first is a simple *wait* period, and the second is a random *interval*. The purpose of the wait period is to allow time for responses by the representatives to traverse the group thereby suppressing non-representative feedback. At the same time, they cannot be too long, so that losses not covered by the current representative set can be detected in a timely fashion. The purpose of the random interval is to space out feedback responses and allow probabilistic suppression to reduce the amount of feedback.

The wait interval is set as percentage of the estimated GRTT described below.

4.5.1 GRTT Measurement

Computing the maximum group round-trip (GRTT) time is a difficult and ambiguous proposition. If we simply keep track of the worst RTT in the group we can get an overly pessimistic value, i.e., a value inordinately large due to transient congestion. Once a large value has been established, it might never be reduced. The obvious answer is to decay the value over time. Again a problem arises in in choosing what decay function to use.

Another alternative is to measure all RTTs and use a simple averaging filter. One then winds up with an average which is not desirable. If we decay only the worst values, then we have to define a maximum interval over which to wait for responses, which again, is not what we want.

Our solution is for the source to keep a table of the worst RTTs received. Each table entry contains an RTT measurement and a time-to-live (TTL). Each time a packet is sent, the TTL is decremented. The current TTL values is set to 10. We chose this value to reduce rapid turnover in the RTT set in the case of lost packets. Since we are measuring RTT for every feedback packet received, the RTTs should be updated once for every data packet sent.

We keep track of the three worst RTTs received. When a new RTT is received it is assigned a TTL. Note that we do not generate any additional messages to estimate the GRTT.

4.5.2 Timer Setting

Each receiver's wait period and suppression interval is defined as a percentage of the advertised GRTT. Since ACKs provide little flow control information, they can be suppressed to an arbitrary degree. Ideally we only need one ACK per packet. Currently we set the ACK wait interval to 1 GRTT.

In the case of NACKs, we are faced with the following tradeoff. On one hand we seek to minimize the number of NACKs. On the other, we wish to receive NACKs in a timely fashion since we need to react to them as quickly as possible. In the normal case, representatives should be selected in such a way as to maximize the probability of responding to packet loss. When representatives fail to cover the packet loss, the backup mechanism of non-representatives sending NACKs must react as quickly as possible.

4.6 Overhead

Since our feedback control mechanism does not rely on RTT estimation, the only bandwidth overhead required by our scheme is the distribution of a new representative set and GRTT. GRTT measurements are sent as part of a packet header, and hence do require separate messages.

We tried to keep the computational overhead to a minimum. The most computationally intensive features of our algorithm are in the source. The sizes of the representative set and maximum RTT tables are bounded to limit the amount of computational overhead.

Similarly, the space requirements at the source are bounded by the representative set and maximum RTT table sizes.

5 Results

We use a simple simulator to evaluate our feedback suppression mechanism. The goal is to evaluate the amount and timeliness of the feedback generated by the receivers in a multicast group connected using an arbitrary network topology with loss rates and delays specified per link.

5.1 Evaluation Methodology

5.1.1 Network Topologies

Early tests of the feedback suppression mechanism utilized a simple binary tree network with the source transmitting at the root of the tree and all other nodes being receivers. While such networks were easy to generate and provided useful preliminary information on the feedback suppression mechanism, it was necessary to test the feedback suppression mechanism on more realistic networks.

To generate more realistic topologies we use Steve Hotz's Network Topology Generator [9]. The networks generated were three level networks with low delay links at the first and second levels and large delay links at the third level. This is intended to be a loose approximation of a real internetwork topology with a high speed backbone, slower regional networks and finally slow final links. The loss rates were randomly chosen over the interval 0-5%. See [9] for more details on the generation of the networks.

The receivers are all at the third level and hence will tend to be leaves, but still have different distances to the source. The source is at the top level.

5.1.2 Simulator

The simulator generates a minimum delay spanning tree from a given network topology. Link delays and drop rates are specified per link. No capacity modeling is done, i.e. all links have infinite bandwidth. End-to-end delay is simply the sum of the link delays, and no delay jitter is introduced. The membership of the group is static.

Congestion is modeled in a very simple way. At the beginning of a specified interval, a link is chosen at random, and given a high loss rate. After the specified congestion interval (in number of source packets) has elapsed, the loss rate is returned to normal. In the current simulations, the congestion loss rate is set to 10%, and the interval is set to 20 source packets. Congestion is introduced once per congestion interval.

The source transmits data packets at a constant rate of 10 packets a second. We describe the simulation parameters below and Table 1 summarizes them.

- **Representatives** To measure the effect of representatives, we simulated groups with 0, 3, 5, and 10 representatives.

- **NACK Wait Interval** As explained in Section 4.5, the feedback timer delay has two components. The first is a simple wait interval, and the second is a random interval. The wait interval is a percentage of the estimated maximum group round-trip time (GRTT). The feedback time is scheduled by adding the wait and random intervals.

  In our simulations we used wait times of 0, 25, and 100% of the maximum GRTT.

- **Maximum Link Loss Rate**

  Each link is randomly assigned a value between 0 and the maximum loss rate. The overall loss rate of the network is a function of the link loss rates, network size, and topology.

  In our simulations, the network loss rate is very high. The networks typically have a 9% loss rate since we wanted to test our feedback mechanism on the worst possible cases.

Representatives	0, 3, 5, 10
NACK wait interval	0, 0.25, 1
Maximum link loss rate	0.01, 0.02, 0.05,

Table 1: Simulation Parameters

- **Congestion Interval** The length of time that congestion lasts. At the beginning of the interval, a link is chosen and given a high loss rate. At the end of this interval, the loss rate returns to normal.
- **Congestion Loss Rate** The loss rate on the congested link.

We ran our simulator using a randomly-generated, 650-node topology, with 50, 100, and 500 level-1, level-2, and level-3 nodes, respectively. In each simulation run, we vary one of the parameters in Table 1, while keeping all the other parameters constant.

5.2 Evaluation Metrics

In this evaluation we are concerned with two metrics. The first is the quantity of the feedback, and the second is timeliness.

For feedback quantity, we look at the total amount of feedback received. The less the better. Ideally, we want a constant amount of feedback per packet, independent of group size.

For timeliness, we normalize the feedback time to the maximum GRTT over the entire simulation. Ideally we would like the RTT to be the minimum RTT for a given congested point to the source. In practice, it is difficult to select a representative that is the first representative on the far side of the congested link. If we shorten the random suppression interval, the probability of receiving feedback from the first congested receiver increases, but the amount of feedback is increased. While this is not a problem for leaf links, it can create a great deal of feedback if the congested link occurs near the root of a large multicast tree.

5.3 Evaluation Results

In the graphs shown in this section, each point corresponds to the average of a sequence of 20 simulation runs where all parameters were kept constant. We evaluate the effects of the NACK wait timer, representatives, and group size in the quantity and quality of the feedback generated by the multicast group. The graphs that show amount of feedback generated by the group plot number of packets received at the source against data packet sequence number. The timeliness graphs plot the minimum time for the source to receive feedback normalized to GRTT against data packet sequence number.

5.3.1 NACK Wait Timer Settings

Figures 1 and 2 show how the NACK wait interval influences the quantity and the timeliness of the feedback received at the source. Figure 1 shows the amount of feedback generated by the group in terms of number of NACKs and ACKs, and Figure 2 shows the time for NACKs to arrive at the source normalized to the corresponding GRTT. For these simulations, we use 650-member groups, 3 representatives, and a maximum link loss rate of 1%. Note that this maximum link loss rate resulted in an overall loss rate of approximately 9%, which is a considerably higher loss rate than what is observed in real networks. Therefore, in our simulations, we are driving our feedback control algorithms with pessimistic scenarios in terms of loss rates.

Regarding NACKs, in the case where the NACK wait timer is zero (Figure 1(a)), the average number of NACKs received is at least twice the average number of NACKs when the NACK wait time is 1 GRTT (Figure 1(c)). This is due to the fact that longer NACK wait intervals give representatives a better chance to suppress non-representative NACKs.

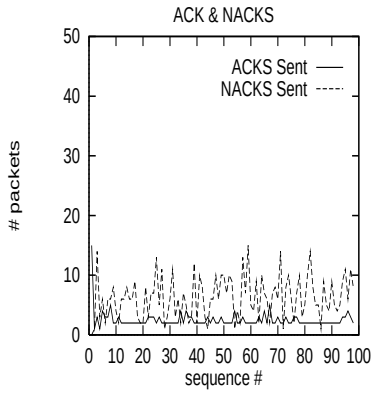
During the first 10 data packets while representatives are first being selected, we receive noticeably more feedback. The representative set is constantly changing during the simulation in response to packet loss being detected in subtrees not covered by the current representative set. This accounts for some of the variation in the amount of feedback generated.

When comparing the graphs obtained for the different NACK timer settings in the 3-representative case with the 0-representative case [5], it is clear that representatives greatly reduce the impact of how the NACK wait timer is set. This is because when there are no representatives, feedback control relies on pure suppression and hence timer setting is crucial. Representatives provide immediate feedback and only rely on NACK timers for packet losses they still do not cover.

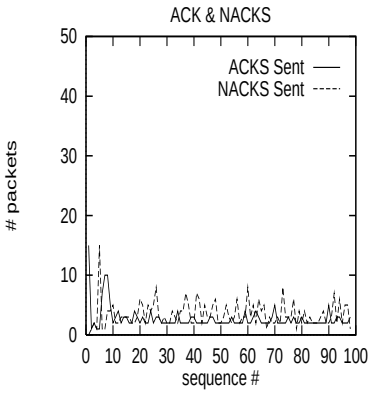
If the current representative set covered all packet losses in their multicast group, longer NACK timers would be the way to go. However, since in real networks congestion occurs at different points at different times, NACK wait timers should be set so that non-representatives can still respond to losses not covered by the current representatives in a timely fashion.

The graphs in Figure 2 show the impact of NACK timers on feedback timeliness. There is the obvious result that shorter NACK timers allow feedback to be received sooner. While in the case of 1 GRTT NACK wait timer, it takes on average 1.5 GRTT for feedback to arrive at the source, it takes less than 1 GRTT when the NACK wait timer is 0. However, as we discuss in the next section, representatives compensate for longer NACK timers by sending immediate feedback.

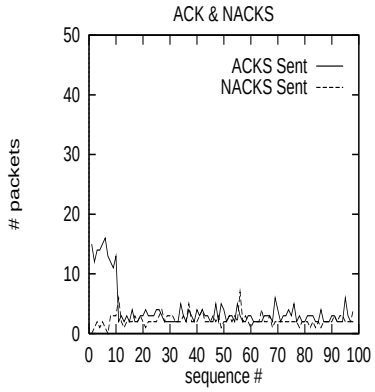
[5] The 0-representative graphs with different NACK timer settings have been omitted due to space limitations.



(a) NACK wait timer is 0% of the maximum GRTT.

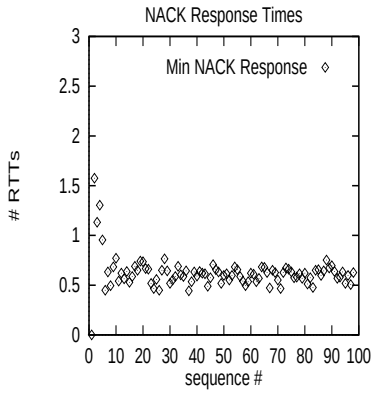


(b) NACK wait timer is 25% of the maximum GRTT

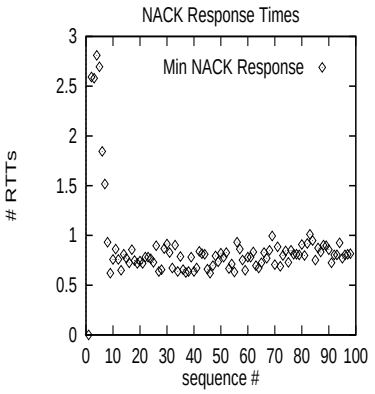


(c) NACK wait timer is 100% of the maximum GRTT

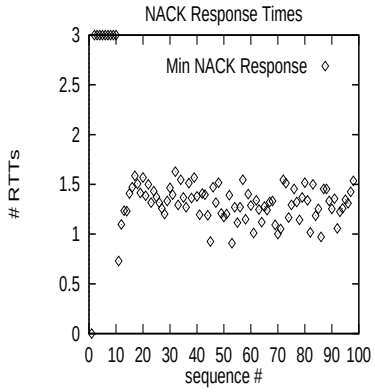
Figure 1: Effects of NACK timers on feedback. Simulation runs used 650-member multicast groups with 3 representatives, maximum link loss rate of 1%, and NACK wait timer equal to 0, 25, and 100% of the GRTT, respectively. Network loss rate is approximately 9%.



(a) NACK wait timer is 0% of the maximum GRTT



(b) NACK wait timer is 25% of the maximum GRTT



(c) NACK wait timer is 100% of the maximum GRTT

Figure 2: Effects of NACK timers on feedback. Simulation runs used 650-member multicast groups with 3 representatives, maximum link loss rate of 1%, and NACK wait timer equal to 0, 25, and 100% of the GRTT, respectively. Network loss rate is approximately 9%.

During the first 10 data packets, we also observe the impact of the representative set startup on feedback timeliness.

5.3.2 Representatives

Figures 3 and Figure 4 show the amount and timeliness of feedback for different representative set sizes. These simulations used 650-member groups, a maximum link loss rate of 1%, or approximately 9% overall loss rate, and NACK wait interval of 25% of GRTT. Graphs (a), (b), (c), and (d) in each figure correspond to representative set sizes of 0, 3, 5, and 10, respectively.

Figure 3 shows how representatives can reduce the amount of feedback received. We observe that using representatives result in a significant reduction in the amount of feedback generated. Overall, the 3-representative case (averaging 3 NACKs per packet sent) generates less feedback than the 0-representative case (7 NACKs per packet), but still perform effective suppression. The 10-representative case however seems to be more effective at limiting the variance in the amount of feedback. This is due to the fact that the 10-representative set provides a better coverage of packet losses, and therefore keeps the amount of feedback generated by the group more constrained.

For the non-empty representative set graphs, the amount of feedback is high during the first 10 packets because representatives were still being elected.

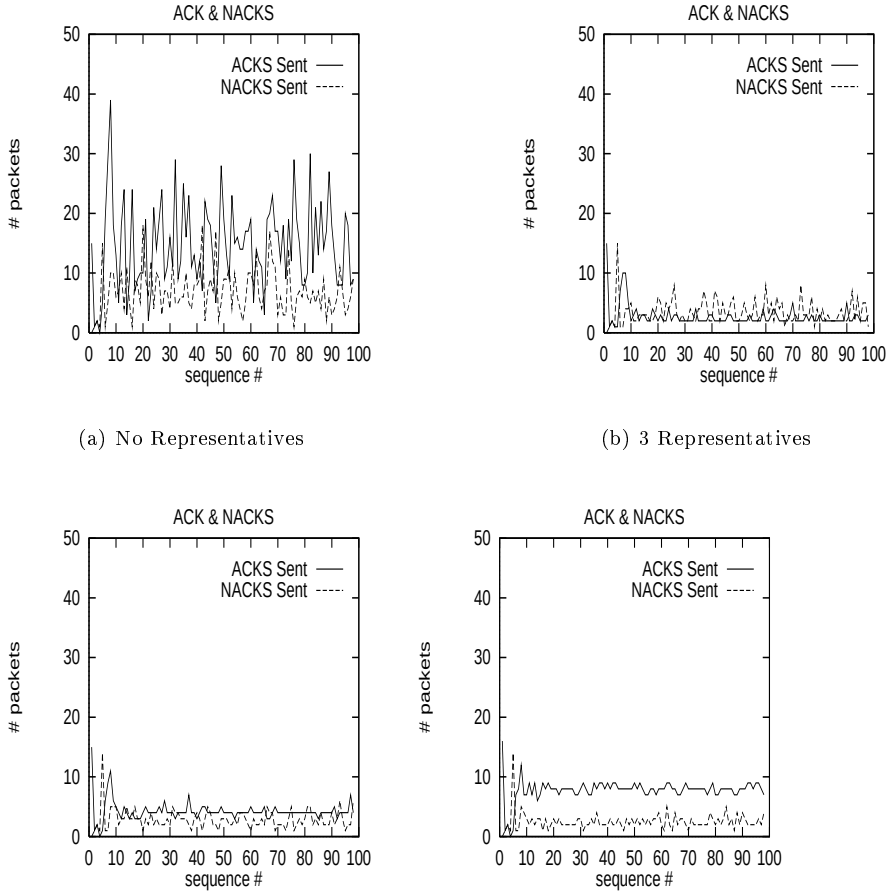
There is a tradeoff between the amount of feedback generated and the timeliness of the protocol. On average, representatives lower the amount of feedback without incurring unreasonable latency in getting feedback to the source. Figure 4(a) shows the non-representative case. A pure suppression scheme with the NACK wait interval equal to 0 results in an improvement of 1/4 GRTT over the case where the representative set in not empty. In other words, representatives pay a penalty of about 1/4 GRTT in terms of timeliness.

In our simple simulation model, representatives tend to be farther from the source. This is due to the fact that the farther down the tree a receiver is, the more likely it is to lose a packet. The more likely a receiver loses packets, the more likely it is that it be elected as a representative. Hence representatives will tend not be the receivers closer to the source.

In this section we showed that for a given multicast group size, we can perform effective suppression with very few representatives. In the next section we show how our algorithm performs for different group sizes.

5.3.3 Group Size

Figures 5 and Figure 6 show the amount and timeliness of feedback generated by groups of different sizes. We use group sizes of 175, 300, and 650, and set the NACK timer at 25% of the GRTT, and the maximum link loss rate at 1%, or 9% approximate overall loss rate. We use a 3-representative set in all cases.



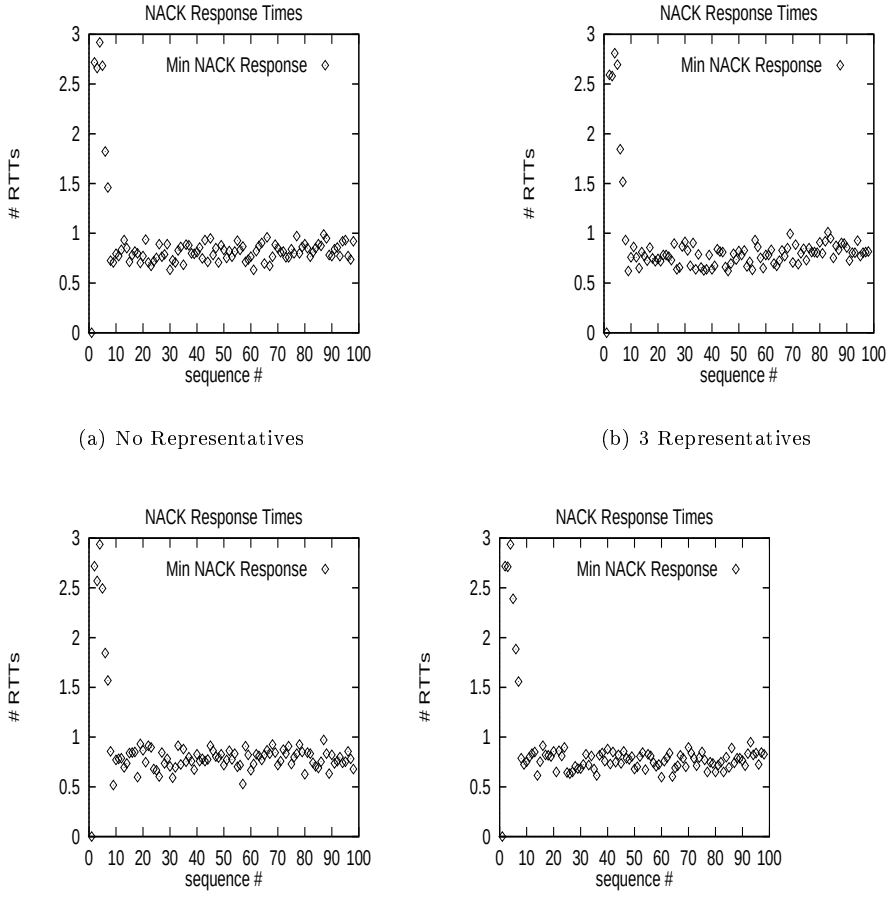
(a) No Representatives

(b) 3 Representatives

(c) 5 Representatives

(d) 10 Representatives

Figure 3: Effects of representatives on feedback. We use 0, 3, 5, and 10 representatives in a 650-member group. NACK wait time is 25% of GRTT, and maximum link loss rate is 1%. Network loss rate is approximately 9%.



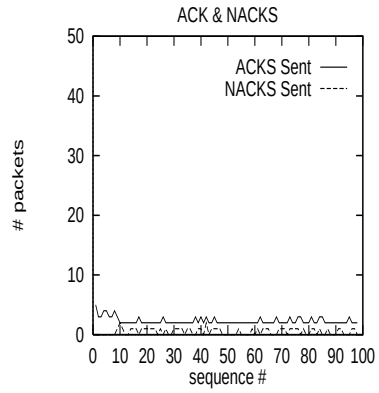
(a) No Representatives

(b) 3 Representatives

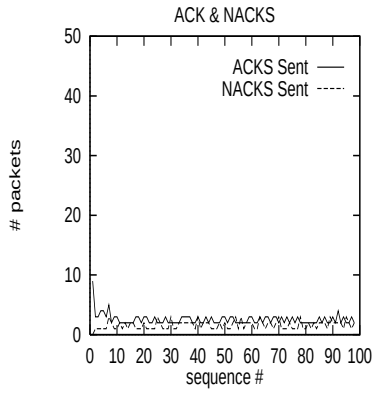
(c) 5 Representatives

(d) 10 Representatives

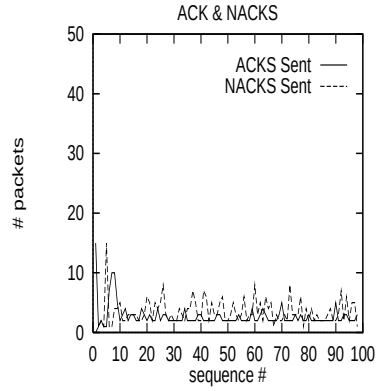
Figure 4: Effects of representatives on timeliness. We use 0, 3, 5, and 10 representatives in a 650-member group. NACK wait time is 25% of GRTT, and maximum link loss rate is 1%. Network loss rate is approximately 9%.



(a) 175-member group



(b) 300-member group



(c) 650-member group

Figure 5: Effects of group size on amount of feedback. We use 175, 300, and 650-member groups. NACK wait time is 25% of GRTT, maximum link loss rate is 1%, and number of representatives is 3. The network loss rate is approximately 9%.

From Figure 5, we observe that the size of the group has relatively little impact on the overall feedback received. This confirms that a few representatives are quite effective in keeping feedback amount low. A larger representative set (5 representatives instead of 3) for the 650-member group (Figure 5(c)) results in less variation in the amount of feedback. As we pointed out in Section 5.3.2, this is due to the fact that more representatives provide a better coverage of packet loss.

Similarly to the amount of feedback generated, Figure 6 shows that feedback timeliness does not degrade as the group size increase. On average, the time for the source to receive feedback is kept under 1 GRTT for all groups sizes shown.

5.3.4 Congestion

We can see from Figure 7 that when congested links are modeled in a multicast tree, the impact of representatives on the amount and variability of feedback is obvious. With no representatives, there is a large variability in the amount of feedback. As representatives are added, the variability in the amount of feedback is greatly reduced. Spikes do occasionally occur which is undesirable. The spikes occur when congestion occurs far up the multicast tree, causing a large number of receivers to schedule NACKS. The suppression mechanism is then not able to suppress as many replies as we would like. The spikes could be reduced by increasing the timers values, but this would also reduce the overall timeliness of the responses. More work is required to address this problem.

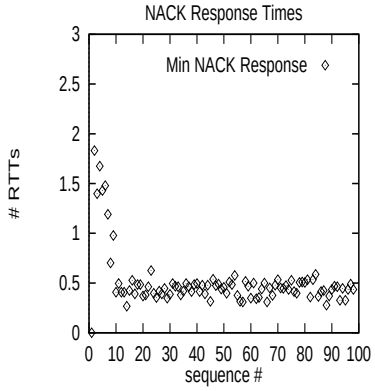
In Figure 8, the timeliness of the NACKs received in the representative case compares favorably with the non-representative case.

6 Future Work

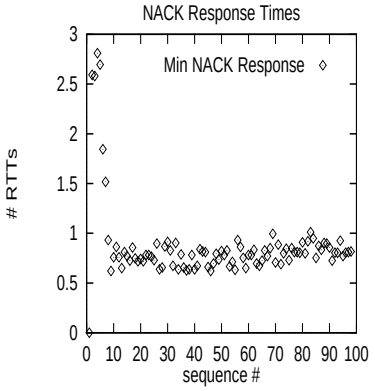
We have evaluated the utility of representatives in purely lossy networks. We expect that performance will be even better in real networks in which losses result primarily from congestion. With this in mind, future simulations need to take into account network congestion and bottleneck links. In the short term, we can extend the current simple simulator to create especially lossy links to simulate congestion and vary these loss rates over time to simulate transient congestion in the network.

An obvious optimization to reduce feedback is to only allow representatives to send ACKs. This will greatly reduce ACKs while still providing protection in the case of congestion collapse. NACK feedback is not affected.

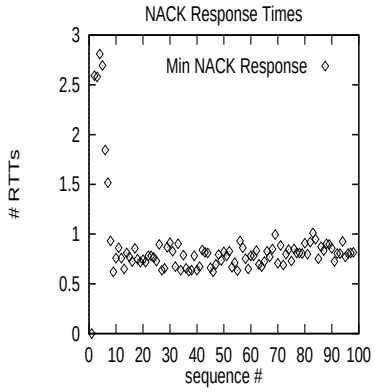
In addition to the random topologies we have examined, we will investigate star networks that one might find in satellite systems, as well as networks containing mobile hosts. We will also investigate the possibility of automatically determining the a good representative set size.



(a) 175-member group

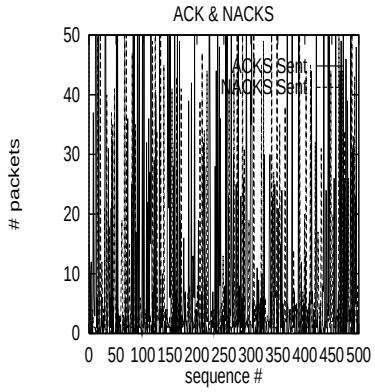


(b) 300-member group

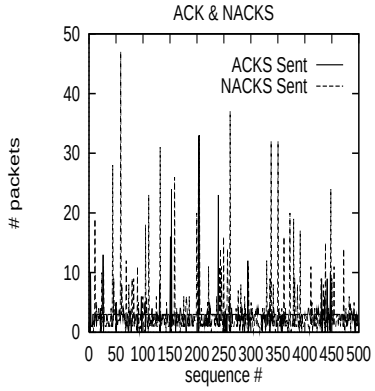


(c) 650-member group

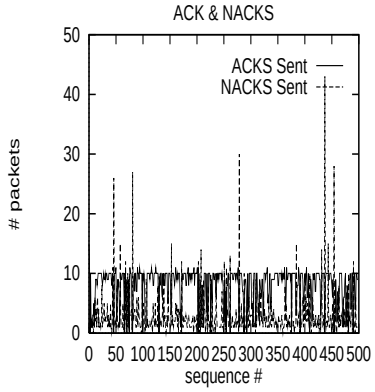
Figure 6: Effects of group size on timeliness of feedback. We use 175, 300, and 650-member groups. NACK wait time is 25% of GRTT, maximum link loss rate is 1%, and number of representatives is 3. The network loss rate is approximately 9%.



(a) No representatives

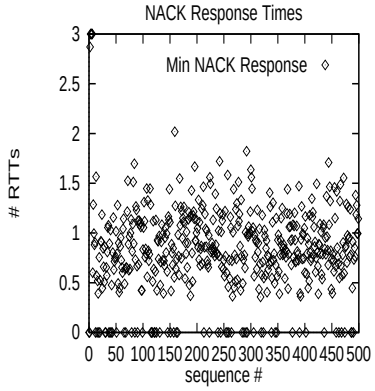


(b) 3 representatives

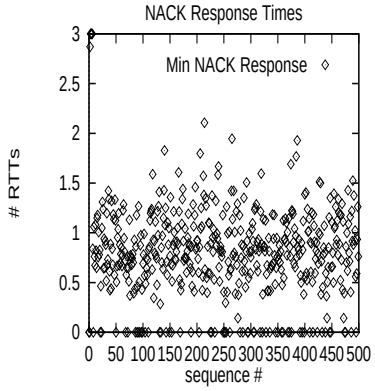


(c) 10 representatives

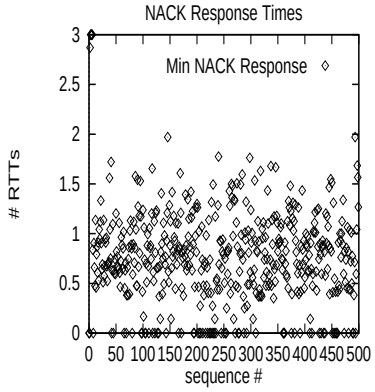
Figure 7: Effects of congestion on the amount of NACK feedback for the 650 member group. Aggregate network loss rate is approximately 9%.



(a) No representatives



(b) 3 representatives



(c) 10 representatives

Figure 8: Effects of congestion on the timeliness of NACK feedback for the 650 member group. Aggregate network loss rate is approximately 9%.

Our goal is to build a full flow control mechanism suitable for bulk data distribution in a multicast environment. Since any flow control scheme must detect congestion in the network, we need to investigate congestion detection schemes. In addition to evaluating the use of packet loss as a congestion metric as is done in traditional TCP, we would also like to investigate delay-based metrics like that used in TCP-Vegas [3].

7 Conclusion

This paper described a multicast feedback control scheme that is part of our efforts to build a scalable reliable, flow-controlled, multicast transport protocol suitable for bulk data transfer applications.

The major benefit of our algorithm derives from the fact that we do not need to compute round trip time from receivers to the source, and we do not require knowledge of group membership or network topology.

We use a small set of representatives in combination with probabilistic suppression to limit feedback, yet not significantly degrading the timeliness of the feedback with respect to the worst round trip time for the group.

We investigated the performance of our feedback control mechanism in purely lossy networks with no capacity modeling. Representatives show a marked improvement over a purely suppression oriented algorithm.

References

[1] J.S. Ahn, P.B. Danzig, Z. Liu, and L. Yan. Evaluation of tcp vegas: Emulation and experiment. *1995 ACM SIGCOMM Conference*, pages 185–195, October 1995.

[2] A.J. Ballardi, P.F. Francis, and J. Crowcroft. Core based trees (cbt). *Proc. of the ACM SIGCOMM'93, San Francisco, CA*, August 1993.

[3] L.S. Brakmo, S.W. O'Malley, and L.L. Peterson. TCP Vegas: New techniques for congestion detection and avoidance. *1994 ACM SIGCOMM Conference*, pages 24–35, May 1994.

[4] D.D. Clark, M.L. Lambert, and L. Zhang. Netblt: A high throughput transport protocol. *1987 ACM SIGCOMM Conference*, pages 353–359, August 1987.

[5] P. B. Danzig. *Optimally Selecting the Parameters of Adaptive Backoff Algorithms for Computer Networks and Multiprocessors.* PhD thesis, University of California, Berkeley, December 1989.

[6] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei. An architecture for wide-area multicast routing. 1994 ACM SIGCOMM Conference, February 1994.

[7] S. Floyd, V. Jacobson, S. McCanne, C.G. Liu, and L. Zhang. A reliable multicast framework for light-weight sessions and application-level framing. *1995 ACM Sigcomm Conference*, pages 342–356, October 1995.

[8] M. Grossglauser. Optimal deterministic timeouts for reliable scalable multicast. *Proc. of the IEEE Infocomm'96, San Francisco, CA*, pages 1425–1432, March 1996.

[9] S. Hotz and R. Nagamati. Network topology generator (NTG): A tool for generating network topology and policy for protocol simulation purposes. Technical Report, Computer Science Department, University of Southern California, Spring 1992.

[10] S.K. Singhal H.W. Holbrook and D.R. Cheriton. Log-based receiver-reliable multicast for interactive simulation". *1995 ACM SIGCOMM Conference*, pages 328–341, September 1995.

[11] V. Jacobson. Berkeley tcp evolution from 4.3-tahoe to 4.3-reno. Proceedings of the British Columbia Internet Engineering Task Force, July 1990.

[12] Van Jacobson. Congestion avoidance and control. *ACM SIGCOMM 88*, pages 273–288, 1988.

[13] T. Turletti J.C. Bolot and I. Wakeman. Scalable feedback control for multicast video distribution in the internet. *Proc. of the Conference on Communications Architectures, Protocols, and Applications, ACM SIGCOMM 1994*, August 1994.

[14] J.B. Nagle. On packet switches with infinite storage. *IEEE Transactions on Communications*, 35(4):435–438, April 1987.

[15] S. Pingali, D. Towsley, and J. F. Kurose. A comparison of sender-initiated and receiver-initiated reliable multicast protocols. *1994 ACM SIGMETRICS Conference*, May 1994.

[16] K.K. Ramakrishnan and Raj Jain. A binary feedback scheme for congestion avoidance in computer networks. *ACM Transactions on Computer Systems*, 2(8):158–181, May 1990.

[17] H. Schulzrinne. A transport protocol for real-time applications. Internet Draft, Internet Engineering Task Force, Audio-Video Transport WG, March 1993.

[18] T. Turletti. H.261 software codec for video conferencing over the internet. INRIA Research Report 1834, January 1993.