A Scalable and Robust Feedback Mechanism for Adaptive Multimedia Multicast Systems

*Alaa Youssef *, Hussein Abdel-Wahab, and Kurt Maly*
Department of Computer Science
Old Dominion University
Norfolk, VA 23529, USA
(youssef,wahab,maly)@cs.odu.edu

Abstract

We present a mechanism for providing state feedback information to multicast sources of multimedia streams in a scalable and robust manner. The presented feedback mechanism is suitable for best-effort unreliable networks such as the Internet. This mechanism is useful for controlling the transmission rate of multimedia sources in both cases of layered and single-rate multicast. It allows for determining the worst case state among a group of receivers, where each receiver may be in one of a set of finite states, and is applicable in receiver-driven as well as in sender-driven adaptive multimedia systems. Simulation results show that the presented feedback mechanism scales well for very large groups of up to few thousands of participants. The efficiency of the proposed mechanism in eliminating the reply implosion problem, its robustness in facing network losses, as well as its responsiveness are illustrated. In addition, the advantages of the proposed mechanism over other feedback mechanisms are demonstrated. Moreover, adaptive enhancements for the mechanism are proposed to maintain its scalability for even larger groups.

Keywords

Feedback, multicast, adaptive multimedia applications.

1 INTRODUCTION

Multimedia streams are becoming a main component of modern distributed collaboration and tele-teaching systems. Most of these systems rely on IP multicasting in order to scale to large groups of participants. However, the quality of service (QoS) requirements of the multimedia streams demand special treatment. The main approaches taken for handling the requirements of

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35388-3_42

H. R. van As (ed.), *High Performance Networking*

multimedia streams can be broadly classified as either proactive or reactive. The proactive approach relies on the existence of a resource reservation protocol (Gupta *et al.* 1995, Zhang *et al.* 1993), and underlying scheduling mechanisms, to reserve and guarantee end-to-end resources. On the other hand, the reactive approach relies mainly on the ability of the application to adapt itself to the level of available resources (Bolla *et al.* 1997, Bolot *et al.* 1994, Cheung *et al.* 1996, McCanne *et al.* 1996). Most of these approaches, for handling multimedia streams, manage individual connections in isolation of others, which may lead to a state of competition for resources among streams belonging to the same session, thus decreasing the overall perceived session quality. Our approach, however, is to dynamically control the QoS offered by the system across the set of connections belonging to the application. This control is based on the application semantics, and focuses on maintaining the best overall quality of session, at every instant during the session lifetime. To this end, we introduced the concept of *Quality of Session (QoSess)* (Youssef *et al.* 1997).

In (Youssef *et al.* 1998), we propose an architecture for a middle-ware platform, which supports collaborative multimedia applications by providing QoSess control mechanisms. Conceptually the QoSess control layer acts as a closed loop feedback system that constantly monitors the observed behavior of the streams, takes inter-stream adaptation decisions, and sets the new operating level for each stream from within its range of permissible operating points. Over wide area network connections, the QoSess control layer manages the resources that are collectively reserved, for the streams of a distributed application, by a resource reservation protocol, such as RSVP. Multi-grade streams are centric to the QoSess framework, in order to support heterogeneity of receivers and network connections. Multi-grade transmission can be achieved either by hierarchical encoding (McCanne *et al.* 1997, Senbel *et al.* 1997), or by simulcast which is the parallel transmission of several streams each carrying the same information encoded at a different grade (Li *et al.* 1996, Willebeek-LeMair *et al.* 1997).

In this paper, we present one of the main building blocks of the QoSess control layer: a scalable and robust state feedback mechanism. This mechanism provides the source of a multimedia stream with deterministic information regarding the state of the receivers. The state of a receiver may be defined as the layers which it is interested in receiving from the source of a hierarchically encoded stream. Given this knowledge, the sender can suppress or start sending the correct layers. The feedback mechanism is not only important for saving the sender's host and LAN resources but for saving WAN resources as well in situations where the application's addressing scheme for the layers does not permit the intermediate routers to suppress unwanted layers, or where the session is conducted over an Intranet whose subnets are inter-connected via low level switches that do not implement the IGMP protocol (Deering *et al.* 1990) for suppressing unwanted multicast packets. Soliciting feedback

from receivers in a multicast group might create a *reply implosion* problem, in which a potentially large number of receivers send almost simultaneous redundant replies. We present a scalable and robust solution to this problem.

The rest of this paper is organized as follows. In Section 2, the role of feedback in several adaptive multimedia multicast systems is illustrated. A brief survey of the different approaches for providing scalable feedback is presented in Section 3. The proposed feedback mechanism is described in detail in Section 4, followed by a performance study and comparison in Section 5. In Section 6, adaptive enhancements for the proposed mechanism in order to support very large groups of receivers are described, and we present our conclusions in Section 7.

2 FEEDBACK ROLE IN MULTIMEDIA MULTICAST

Early attempts towards providing adaptive transport of multimedia streams over the Internet focused on the sender as the entity playing the major role in the adaptation process (Bolot *et al.* 1994, Busse *et al.* 1995). Information about the congestion state of the network, as seen by the receivers, was fedback to the sender which used it to adapt to changes in the network state. In many cases, the monitored performance parameters (e.g., loss rate, delay, jitter, throughput) were mapped, by the receiver, to one of several qualitative performance levels, and reported to the sender (Bolot *et al.* 1994, Busse *et al.* 1995, Cheung *et al.* 1996). The sender adapted its transmission rate by varying the quality of the transmitted media content by means of controlling several encoder parameters (e.g., frame rate, frame size, or quantization step for video streams). The sender often based its decisions on the worst case state reported (Busse *et al.* 1995), and sometimes based it on a threshold of the number of receivers suffering the worst state (Bolot *et al.* 1994). In this approach all receivers have to receive the same quality of multimedia streams regardless of the differences in their capacities and the capacities of the network connections leading to them. Although sometimes it is desired to maintain identical stream quality across all participants of a session (e.g., for some discrete media streams), yet this is not always the case especially with continuous media streams.

The first approach, to address the need for providing a multi-grade service to participants of the same session, was represented by the introduction of the concept of *simulcast* (Li *et al.* 1996, Willebeek-LeMair *et al.* 1997). In a simulcast system, the sender simultaneously multicasts several parallel streams corresponding to the same source, but each is encoded at a different quality level. Each receiver joins the multicast group that matches its capabilities. Within a group, the same techniques of source adaptation, that were mentioned above, are applied within a limited range. Thus, the same feedback mechanisms are also deployed within each group.

With the advent of hierarchical encoding techniques (McCanne *et al.* 1997,

Senbel *et al.* 1997), a new trend in adaptive multimedia transport appeared in which the receiver plays the sole role in adaptation (McCanne *et al.* 1996). In such systems the receiver is responsible for determining its own capabilities, and consequently, it selects the number of layers to receive from the hierarchically encoded stream. The source, however, is assumed to be constantly multicasting all the layers.

While it is very obvious that the layered encoding approach is more efficient in the utilization of resources relative to the simulcast approach, yet it is still debatable whether layered encoding techniques will be able to provide the same media quality as the simulcast encoders which operate in parallel, each optimized for a particular target rate. In spite of this debate, the layered approach is the most appealing from the networking point of view, due to its efficient utilization of network resources, especially bandwidth. However, this approach as described is not as efficient as can be. The fact that the source keeps sending at full rate, all layers, constantly, may lead to the waste of more resources than with simulcast, in the case where no receiver subscribes to some of the layers. On the other hand, augmenting this approach with a simple scalable feedback mechanism that provides the source with information regarding which layers are being consumed and which are not, yields more efficiency in resource consumption, as the sender can get actively involved in the adaptation process by suppressing the unused layers.

The introduction of such a feedback mechanism, for receiver-oriented layered transport of multimedia streams, is not only an added efficiency feature for such transport protocols, but it is also a critical feature for the success of collaborative multimedia sessions in which multiple streams are concurrently active. In such collaboration sessions, multiple streams are typically distributed to all participants of the session, and the overall session quality is determined by the quality of each of the streams as well as by their relative importance and contribution to the on-going activity. In presence of scarce resources, it is logical to sacrifice the quality of one low priority stream for the sake of releasing resources to be used by a higher priority stream. Should the low priority stream source keep pushing all unused layers to the network, the decision taken by the receivers to drop these layers for releasing resources is rendered almost useless. This uselessness will hold true forever for the sender's host and LAN, while the rest of the network may eventually have these resources released as the multicast routers stop forwarding the unused layers. In situations were the application's addressing scheme for the layers does not permit the intermediate routers to suppress unwanted layers, WAN resources may also be wasted.

In the former case, besides the unnecessary delay in releasing resources, the fact that the sender's host and LAN will always be overloaded is very critical, as the session participants on this LAN may not be able to receive other higher priority streams. The problem is more crucial for Intranet based

collaboration systems since all the session participants (senders and receivers) are typically within a few hops from one another (Maly *et al.* 1997).

Moreover, since the sender may be sending only a subset of its layers, it needs to know about the existence of clients for higher layers that are currently suppressed, as soon as these clients subscribe to these layers. This information must be provided to the sender in a timely and scalable way that avoids potential implosion problems in such cases when many clients subscribe to higher layers almost simultaneously. This is likely to happen when some streams are shutdown releasing resources that can be utilized by other active streams.

From the above we conclude that a feedback mechanism is necessary for involving the sender in the adaptation process for receiver driven layered multicast of multimedia streams, especially in the context of collaborative multimedia sessions. Moreover, such a feedback mechanism is essentially the same as, and can replace, feedback mechanisms for supporting simulcast and single-rate multicasts. In the following section, we briefly describe the different approaches to providing scalable feedback, then in Section 4, we introduce the proposed scalable and robust mechanism for providing feedback in adaptive multimedia multicast systems.

3 EXISTING SCALABLE FEEDBACK TECHNIQUES

Soliciting information from receivers in a multicast group might create a *reply implosion* problem, in which a potentially large number of receivers send almost simultaneous feedback messages that contain redundant information. Typical solutions to address this problem include *probabilistic reply*, *expanding scope search*, *statistical probing*, and *randomly delayed replies* (Bolot *et al.* 1994).

Probabilistic reply: In a probabilistic reply scheme, a receiver responds to a probe from the source with a certain probability. If the source does not receive a reply within a certain timeout period, it sends another probe. This scheme is easy to implement. However, the source is not guaranteed to receive the worst news from the group within a certain limited period. In addition, the relationship between the reply probability and the group size is not well defined.

Expanding scope search: In the expanding scope search scheme, the time-to-live (TTL) of the probe packets sent by the source is gradually increased. This scheme aims at pacing the replies according to the source capacity of handling them, since the source does not re-send the probe with increased scope until it has processed all previous replies. Clearly this is efficient only in the case where the receivers are uniformly distributed in TTL bands, which may not be the case.

Statistical probing: This scheme relies on probabilistic arguments for scalability. At the start of a round of probes (called *epoch*), the sender and each of the receivers generate a random key of a fixed bit length. In each probe, the source sends out its key together with a number specifying how many of the key digits are significant. Initially, all digits are significant. If a match occurs at a receiver then that receiver is allowed to send a response. If no response is received within a timeout period, the number of significant digits is decreased by one and another probe is sent. In (Bolot *et al.* 1994), it was shown that there is a statistical relationship between the group size and the average round upon which a receiver first matches the key. This scheme is efficient in terms of number of replies needed to estimate the group size. However, as shown in (Bolot *et al.* 1994), the maximum response time (the time needed for the source to identify the worst case of all receivers) is equal to 32 times the worst case round trip time. For a typical worst case RTT of 500 milliseconds, it may take up to 16 seconds to find the worst case state of all receivers.

Randomly delayed replies: In the randomly delayed replies scheme, each receiver delays the time at which it sends its response back to the source by some random amount of time. Clearly, the success of this scheme in preventing the *reply implosion* problem depends to a great extent on the duration of the period from which random delays are chosen. However, the scheme is very appealing, in the sense that it allows for receiving responses from all the receivers in the group, if the delay can be adapted using some knowledge of the size of the group.

From the above basic mechanisms, the *randomly delayed replies* approach, augmented with suppression of redundant replies and careful selection of delay periods, is the most appealing for two main reasons: first, a response is always guaranteed; and second, the response time is expected to be always low. This is the basic idea deployed in IGMP (Internet Group Management Protocol) (Deering *et al.* 1990). In IGMP, the probe is sent to a local area network (LAN), and hence as soon as one of the receivers responds to the probe it is guaranteed that all the other receivers will hear that response and suppress their replies. Also, in such a local environment, the timeout period can be set to a fixed small value. In contrast, in our case, the group of receivers may be distributed over a wide area network (WAN), thus a reply sent by one receiver may not be heard by another before the other one emits its own reply which may be redundant. This implies the need for careful selection of the delay randomizing functions.

A closely related, but different, problem is the negative acknowledgment (NAK) implosion problem associated with reliable multicasting. A solution for the NAK implosion problem, which is based on randomly delayed replies with suppression of redundant NAKs, is adopted by the SRM protocol (Floyd *et al.* 1995). In SRM, when a receiver detects a lost packet, it randomizes

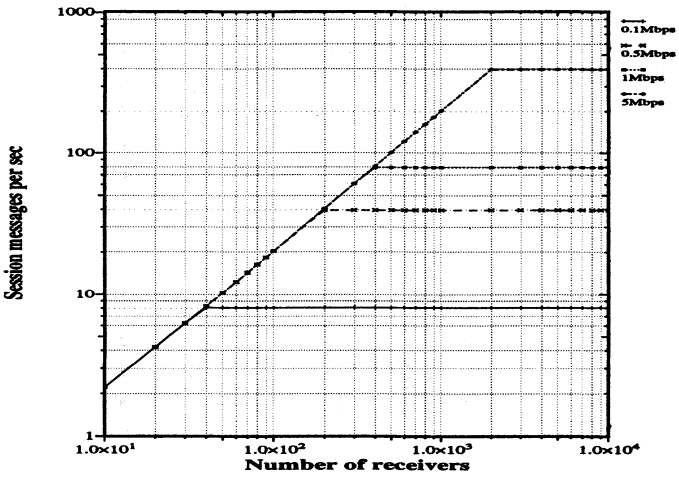


Figure 1 Overhead of session messages

the delay before sending its NAK in the interval $[C_1 d_i, (C_1 + C_2)d_i]$, where d_i is the distance from receiver i to the source, $C1$ and C_2 are constant parameters. Both the NAK and the state feedback implosion problems are similar in the need for soliciting replies from a potentially very large group of receivers. However, with NAKs, whenever a data packet is lost on a link, all the receivers that the faulty link lead to will eventually detect the loss and send a NAK. Thus the distance between a receiver and the faulty link is the major factor that determines when the receiver will detect the fault, and consequently favoring closer receivers, by letting them send their NAKs earlier, implies suppression of more redundant NAKs. On the other hand, in the state feedback problem, the capacity of the receiver, and consequently its state, may not be related to its distance from the source. Therefore, a different criteria for randomizing the delays is required.

In SRM, each receiver must determine its distance from the source to use it in the delay function. The overhead of session messages (typically RTCP reports (Schulzrinne *et al.* 1996)) which are needed for that is not negligible. Figure 1, shows the overhead of RTCP reports for different session sizes and rates, assuming a single source. One of the objectives of the proposed feedback mechanism is to eliminate this high overhead, by designing the mechanism in a way that is not dependent on periodic session messages.

4 A SCALABLE FEEDBACK MECHANISM

In this section, we describe the proposed mechanism for eliciting feedback information from the receivers in a multicast group. The objective of the algorithm is to find out the worst case state among a group of receivers. The definition of the worst case state is dependent upon the context in which the feedback mechanism is applied. It can be the network congestion state as seen by the receivers. This may be useful for applications where a similar consistent view is required for all the receivers, and the source is not capable of providing a multi-grade service, and hence must adapt to the receiver experiencing the worst performance. Another definition, of worst case state as seen by all receivers, is identifying the highest layer a receiver is expecting to receive in a hierarchically encoded stream. This allows the sender to adjust its transmission rate in order not to waste resources on layers that no receiver is subscribing to, and to start sending previously suppressed layers as soon as receivers subscribe to receive them. This is particularly important in the context of managing multimedia streams in collaborative sessions, because in such sessions the sender of a stream is typically simultaneously receiving multiple streams, and hence the assumption that the sender has abundant resources is not valid.

In the rest of the paper, we assume that at every instant in time each receiver is in one state s, where $s = 1, 2, ..., H$. H is the highest or worst case state, and the state of a receiver may change over time.

We consider the general case when neither the group size nor the round-trip time from the sender to each receiver is known. As will be shown later, this information is not necessary as the mechanism estimates the average round trip time in the group, and uses it to adjust its timeout periods.

In the proposed mechanism, the sender sends one type of probe messages, called *SolicitReply* messages, on a special multicast group which the sender and all the receivers join. The probe message contains a RTT field, which contains an estimate for the average round trip time from the sender to the group members. Upon receiving the *SolicitReply* probe, a receiver sets a timer to expire after a random delay period which is drawn from the interval

$$\left[C_1 f(s)\frac{RTT}{2}, (C_1 f(s) + C_2 g(s))\frac{RTT}{2}\right],$$

where $f(s)$ and $g(s)$ are two non-increasing functions of the state s, C_1 and C_2 are two parameters whose values are discussed later in detail. The receiver then keeps listening to the multicast group. If the timer expires, the receiver multicasts a reply message to the whole group. The reply message contains the state information as seen by this receiver (e.g., highest layer expected to receive in a hierarchically encoded stream). On the other hand, if the receiver receives another receiver's reply before its timer expires and that reply

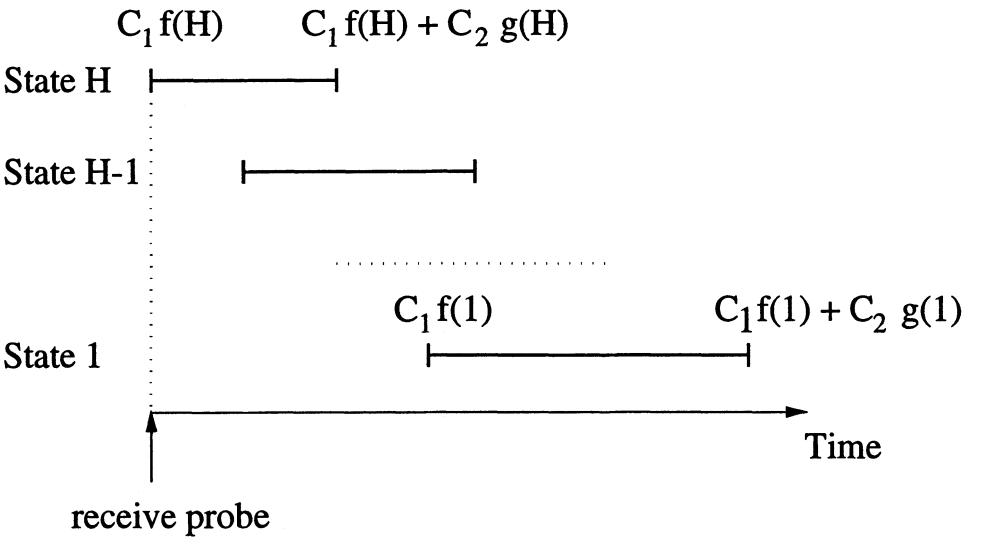


Figure 2 Distribution of timeout periods according to receiver state

contains either the same or higher (worse) state, then the receiver cancels its timer and suppresses its own reply. This implies the need for careful selection of $f(s)$, $g(s)$, C_1, and C_2 in order to avoid the reply implosion problem, while maintaining a low response time. In the subsequent subsections, we discuss in detail choices for $f(s)$, $g(s)$, C_1, C_2, and RTT.

4.1 Selecting the timeout functions

The objective of setting the timeout periods as a function of $f(s)$, and $g(s)$ is to distribute the timeouts as in Figure 2. Receivers in higher states randomize their timeouts over periods that start earlier than receivers in lower states, thus allowing for higher state responses to suppress lower state responses. In addition, the lower state receivers randomize their timeouts over longer periods relative to higher state receivers. This is because as time elapses and no responses are generated this means that the distribution of receivers over states is biased and more receivers belong to the lower states. Thus it is desired to randomize these condensed replies over longer periods.

In order to meet these objectives, $f(s)$ and $g(s)$ must be non-increasing functions of s. Also, $f(H)$ should equal 0 to avoid unnecessary delays in response time, while $g(s) > 0$ must be satisfied for all values of s to allow for randomization of timeout periods. We chose to make $f(s)$ and $g(s)$ linear functions in s in order to avoid excessive delays in response time, where $f(s) = H - s$, and $g(s) = f(s) + k = H - s + k$.

The parameters C_1 and C_2 scale the functions $f(s)$ and $g(s)$. C_1 controls

the aggressiveness of the algorithm in eliminating replies from lower state receivers, while C_2 controls the level of suppression of redundant replies from receivers in the same state. The values of these two parameters are explored in depth in the following sections. The value of k is set to 1. Selecting the value of k is not critical, since the parameter C_2 scales $g(s)$, and the value of C_2 can be tuned to optimize the performance of the mechanism given the selected value of k.

4.2 Exploring the parameter space

In this section, we attempt to find bounds for the ranges of operation of the parameters C_1 and C_2. Obviously, low values for C_1 and C_2 are desired in order to reduce the response time. On the other hand, excessive reduction in the value of either of the two parameters may lead to inefficiency in terms of the number of produced replies possibly leading to a state of reply implosion.

In order to effect a shift in the start time of the timeout periods based on the state of the receiver, as in Figure 2, $C_1 > 0$ must be satisfied for all $s < H$. This shift allows for the high state replies to suppress low state replies. Similarly, $C_2 > 0$ must be satisfied for all values of s, in order to allow for randomization of timeout periods for receivers belonging to the same state, thus enabling suppression of redundant replies which carry the same state information.

To further bound the values of C_1 and C_2, we analyze two extreme network topologies, namely: the chain and the star topologies. Given a certain distribution of receiver distances from the sender, the feedback mechanism exhibits worst case performance when the receivers are connected in a star topology with the sender at its center. This is because connecting those receivers in a star topology maximizes the distance between any pair of receivers, to the sum of their distances from the sender, and hence minimizes the likelihood of suppression of redundant replies. On the contrary connecting those receivers in a chain topology minimizes the distance between any pair, to the difference between their distances from the sender, and hence maximizes the likelihood of suppression of redundant replies. Therefore, for a given distribution of distances, and an arbitrary topology, the performance of the feedback mechanism lies somewhere in between the chain and the star cases.

(a) Chain topology

In the chain topology, the sender is at one end of a linear list of nodes. The rest of the nodes in the list are receivers. Let $r = \frac{RTT}{2}$ be a bound on the one way distance from the sender to any of the receivers or vice versa. Let the sender send a probe at time t. The farthest receiver receives the probe at time $t + r$. If this receiver is the only one in the highest state, and if it emits its reply as soon as it receives the probe, then all other receivers will have heard

this reply by time $t + 2r$. In order to suppress all replies from lower state receivers in this case, $C_1 \geq 2$ must be satisfied. $C_1 = 2$ makes the difference between the start time of two successive states equal to $2r$.

(b) Star topology

In the star topology, the sender is connected to each receiver by a separate link. Any message sent from one receiver to another passes through the sender's node. Let all the receivers be at a distance $r = \frac{RTT}{2}$ from the sender. Thus the distance between any two receivers is equal to $2r$.

Let G_s be the number of receivers in state s, and let T_s be the first timer to expire for receivers in state s. The expected value of T_s is $(C_1 f(s) + \frac{C_2 g(s)}{G_s})r$, since G_s timers are uniformly distributed over a period of $C_2 g(s)r$.

For receivers having the same state, if the first timer expires at time t, then all the timers that are set to expire in the period from t to $t + 2r$ will not be suppressed, and all those that are set to expire after $t + 2r$ will be suppressed. Therefore, the expected number of timers to expire is equal to 1 plus the expected number of timers to expire in a period of length $2r$, which is equal to $1 + \frac{2G_s}{C_2 g(s)}$. Looking at the case of $s = H$, since $g(H) = 1$, then setting C_2 to any value less than 2 does not allow for suppression of any of the redundant replies from receivers in state H. Thus $C_2 > 2$ must be satisfied.

In order to suppress all replies from receivers in state $s - 1$, we must have

$$\begin{array}{lcl} T_s + 2r & \leq & T_{s-1}, \\ (C_1 f(s) + \frac{C_2 g(s)}{G_s})r + 2r & \leq & (C_1 f(s-1) + \frac{C_2 g(s-1)}{G_{s-1}})r, \\ \frac{g(s)}{G_s} - \frac{g(s-1)}{G_{s-1}} & \leq & \frac{C_1 - 2}{C_2}. \end{array}$$

For values of G_s and G_{s-1} which are relatively larger than $g(s)$ and $g(s-1)$, we get $C_1 \geq 2$, which is the same condition for C_1 which we obtained from the chain topology. In Section 5, we explore the effect of C_2 on the performance of the feedback mechanism using simulation experiments.

4.3 Estimating the round trip time

To compute the average round-trip time from the sender to the group of receivers, every probe sent is time-stamped by the sender. That time-stamp is reflected in the reply message together with the actual delay period that the receiver waited before replying. This allows the sender to compute the round-trip time to this receiver. The smoothed average round-trip time, *srtt*, and the smoothed mean sample deviation *rttvar* are computed from the received round-trip time samples, using the same technique applied in TCP (Jacobson

1988), as follows:

$$srtt = \alpha\ srtt + (1 - \alpha)\ sample\ , \qquad \alpha = 7/8\ ,$$
$$rttvar = \beta\ rttvar + (1 - \beta)\ |srtt - sample|\ , \qquad \beta = 3/4\ .$$

In TCP, the amount $srtt + 4\ rttvar$ is used in setting the retransmission timeouts in place of twice the round-trip time. As will be shown in Section 5, this amount is conservative and over estimates the average round-trip time to the group members. Instead we use only *srtt* as the estimate for average round-trip time. The recent value of *srtt* is carried in the *RTT* field of the next probe.

5 SIMULATION STUDY AND PERFORMANCE COMPARISON

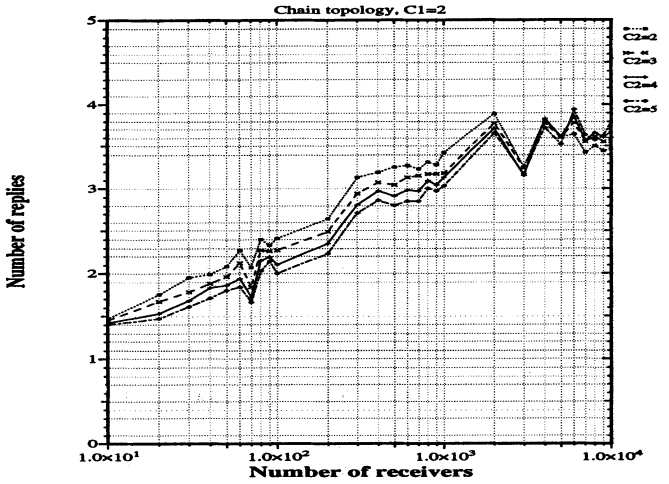
In this section, we examine various issues, related to the performance and tuning of the feedback mechanism, using simulation. First we show the ability of the new feedback mechanism to eliminate the reply implosion problem as we explore the effect of C_2 on its performance. Then we examine the accuracy of the round trip time estimation algorithm. Finally, we further illustrate the scalability and robustness of the proposed feedback mechanism by contrasting it to an alternative candidate mechanism for feedback.

In order to address these issues, we ran several simulation experiments. Each experiment was setup as follows. The group size, G, and the maximum round trip time, RTT_{max}, were selected. Round trip times uniformly distributed in the interval $[0,\ RTT_{max}]$ were assigned to all the receivers, except the worst case state receivers whose round trip times were uniformly distributed in the interval $[t.RTT_{max},\ RTT_{max}]$, for investigating the effect of t over the performance, where $0 \leq t \leq 1$. The number of states, H, was set to 5, and each receiver was randomly assigned one of these states. The choice of 5 states (or layers) is reasonable as the state of the art hierarchical video encoders typically provide a number of layers in this range (McCanne *et al.* 1996, Senbel *et al.* 1997). Also, in applications where feedback information represents the perceived quality of service, typically 3 to 5 grades of quality are used (Bolot *et al.* 1994, Busse *et al.* 1995). The feedback mechanism was simulated under the two extreme network topologies; the chain and the star.

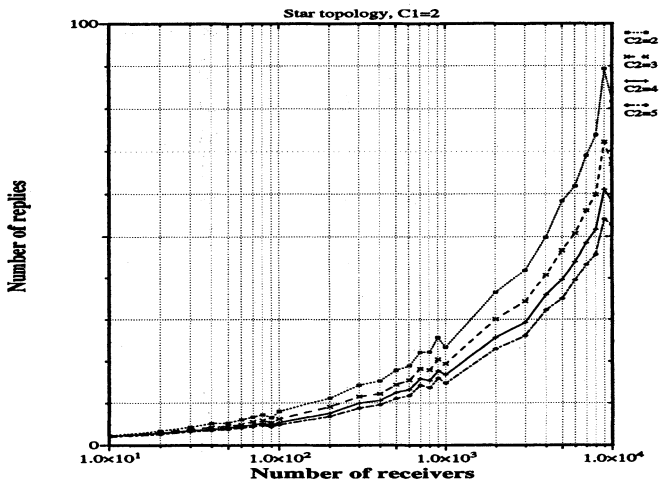
5.1 Bounding C_2

From the analysis in Section 4.2, we obtained the two conditions $C_1 \geq 2$ and $C_2 > 2$. Setting C_1 to its minimum value 2 eliminates replies from lower states, while avoiding unnecessary delays in response time. However, selecting an appropriate value for C_2 is not as easy as such.

In Figure 3, the average number of replies is plotted for different values of



(a) chain topology



(b) star topology

Figure 3 The effect of C_2 on the number of replies

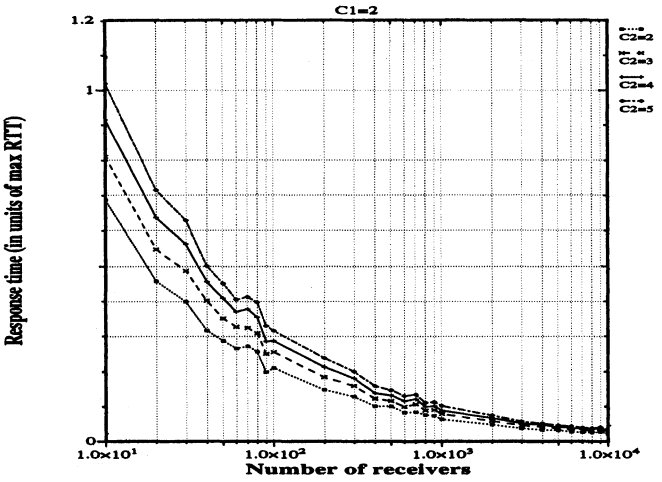


Figure 4 The effect of C_2 on response time

C_2. The value of C_1 was set to 2, for all the experiments in this section, and the average round-trip time was used in the *RTT* field of the probe messages. It is clear from the figure that the performance of the feedback mechanism is not sensitive to the value of C_2 in the case of the chain topology. Also, the figure shows that the reply implosion problem is totally eliminated. Moreover, over 95% of the redundant replies were correct replies (i.e., worst case state replies) which shows the robustness of the mechanism in facing network losses and its efficiency in eliminating non-worst case replies. This also means that, practically, the sender may safely react according to the first received reply. Figure 4 depicts the corresponding average response times. The response time is measured at the sender, and represents the time from sending a probe until receiving the first correct reply. The response time behavior is the same for both topologies because it is dependent on the round-trip times distribution rather than on the topology. As shown in the figure, it is bounded from above by the maximum round-trip time to the group members.

These figures suggest that $C_2 = 4$ is a reasonable setup. $C_2 > 4$ does not significantly reduce the number of replies, while the response time increases. As can be seen from the figures, for typical sessions with up to 100 participants (e.g., IRI sessions (Maly *et al.* 1997)), less than 10% of the receivers reply to a probe, in the worst case, while for larger sessions of thousands of participants the reply ratio is below 1.5%.

5.2 Evaluating the round trip time estimation technique

As mentioned in Section 4.3, the amount *srtt* + 4 *rttvar* is used in setting the retransmission timeouts in place of twice the round trip time, in TCP. Figures 5(a) and (b) compare this approach to using only *srtt* as the estimate for average round trip time. We chose to avoid the conservative approach of TCP, and to use only *srtt*, to avoid unnecessary prolonging of delay periods thus avoiding excessive delays in response time.

5.3 Performance comparison

Here, we further illustrate the scalability and robustness of the proposed feedback mechanism by contrasting it to an alternative candidate mechanism for feedback. The alternative mechanism uses the same approach taken by SRM (Floyd *et al.* 1995) for discriminating between receivers in setting their timeout periods based on their individual distances from the source (i.e. timeouts are selected from the interval $[C_1 d_i, (C_1 + C_2) d_i]$ where d_i is the one way distance from receiver i to the source). This, in turn, depends on the existence of session level messages for the distance estimation process as explained in Section 3.

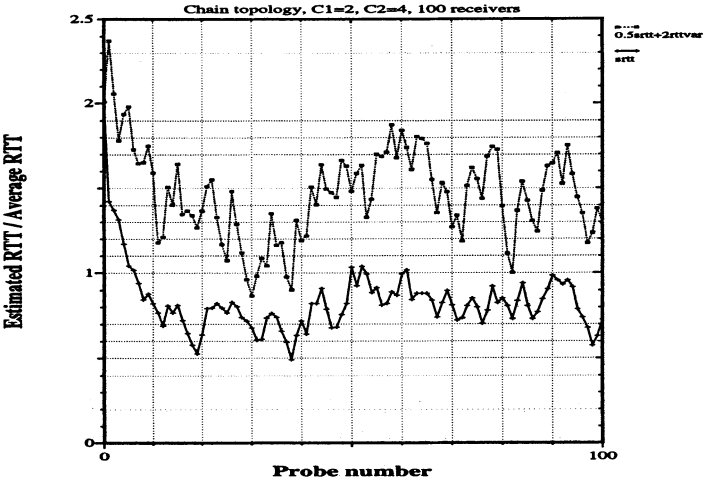
Figures 6 through 8 contrast the performance of our proposed feedback mechanism, A_1, to the alternative feedback mechanism, A_2. The comparisons are performed in two cases. In the first case, the worst case state receivers were distributed at distances in the range [0, RTT_{max}] (i.e., t=0). In the second case, the worst case state receivers were distributed at distances in the range [$0.2RTT_{max}$, RTT_{max}] (i.e., t=0.2).

Figure 6 shows that the total messages sent in response to a probe in the case of the new feedback mechanism, A_1, is much lower than the total response plus session messages for the alternative feedback mechanism, A_2. As discussed in Section 3, the session overhead for A_2 is dependent on the session bandwidth; we depict the two cases of 1Mbps and 5Mbps sessions. For A_2, the session overhead assumed that an epoch (the time span from sending a probe until receiving the last possible reply) will take at most one second.

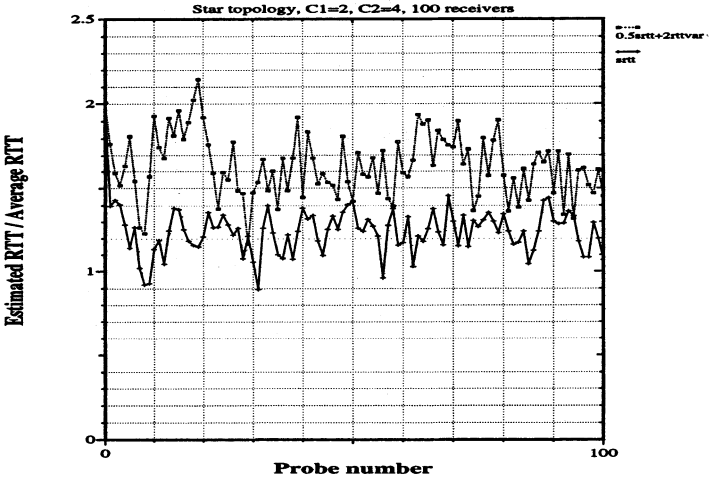
Figure 7 shows that the number of messages carrying correct worst case state information constitute almost all the total messages sent in the new algorithm A_1. In A_2, on the contrary, almost all the messages sent are overhead messages. This demonstrates the robustness of the new feedback mechanism and its tolerance to losses in reply messages.

However, Figure 8 shows that the response time of A_2 is lower on the average. Nevertheless, this is not always the case for A_2, as a slight shift in the distribution of receiver distances reverses this situation and makes the response time of A_1 lower. This trend continues as t increases.

From these charts, we conclude that A_1 is much more robust than A_2. Also,

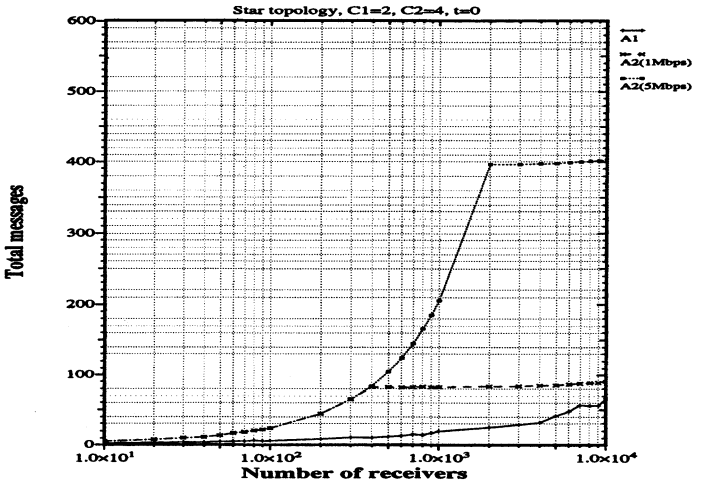


(a) chain topology

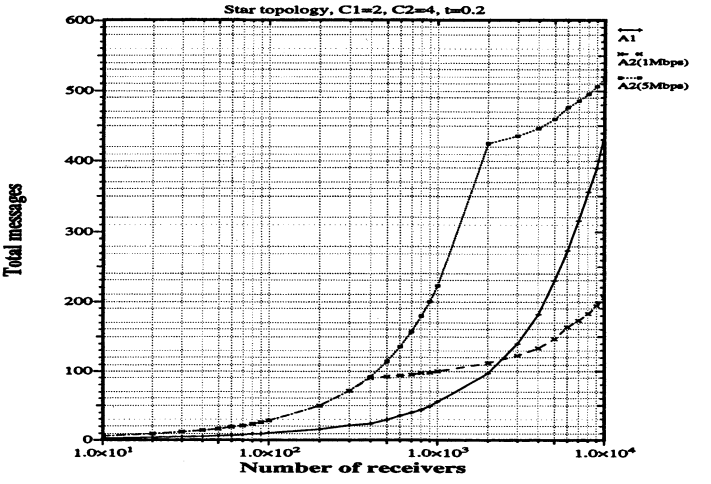


(b) star topology

Figure 5 Accuracy of RTT estimate

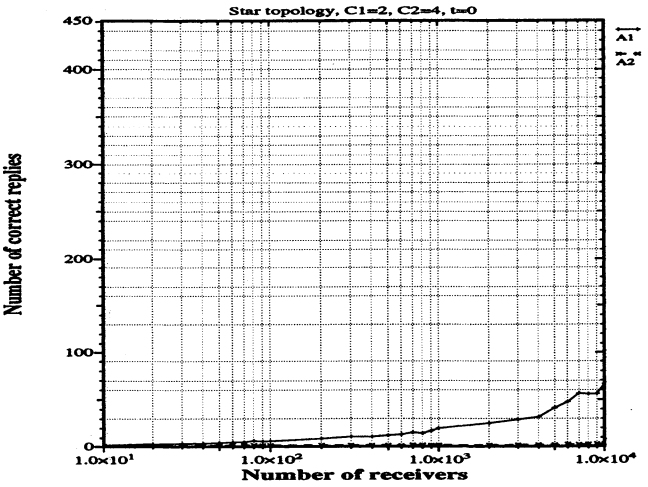


(a) t=0

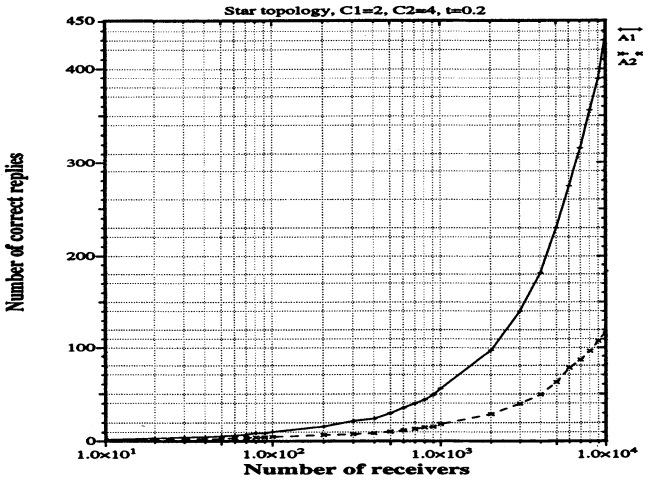


(b) t=0.2

Figure 6 Comparing total messages for the star topology

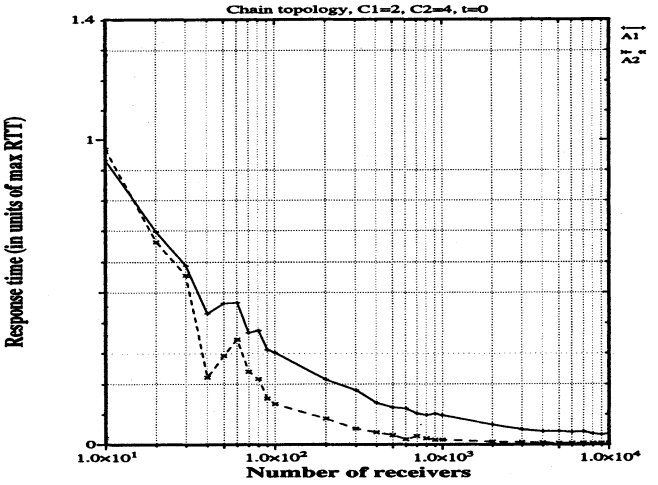


(a) t=0

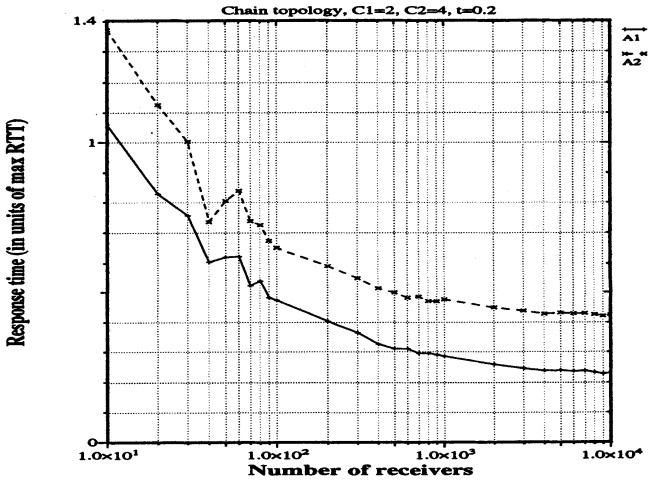


(b) t=0.2

Figure 7 Comparing correct replies for the star topology



(a) t=0



(b) t=0.2

Figure 8 Comparing response time for the star topology

the total overhead of A_1 is always lower than that of A_2 up to sessions of few thousand participants. However, for very large sessions approaching 10000 participants, and for certain distributions of distances of receivers, the overhead of A_1 starts to rise significantly. This is true for star topologies which represent worst case performance for $A1$. For chain topologies, the performance of the algorithm was found to be significantly less dependent on the value of t. In the next section, we address the issue of enhancing the performance of A_1 for very large sessions, and degenerate receiver distributions.

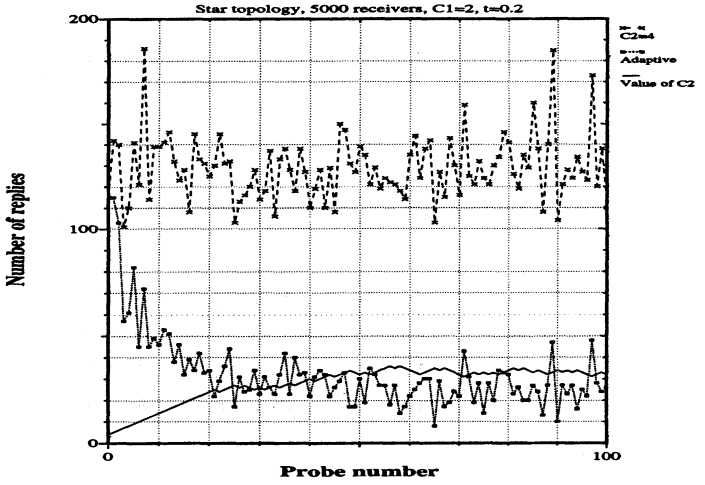
6 ENHANCING THE FEEDBACK MECHANISM

In this section, we present two enhancements for the feedback mechanism. These enhancements further improve the scalability of the feedback mechanism and reduce its overhead.

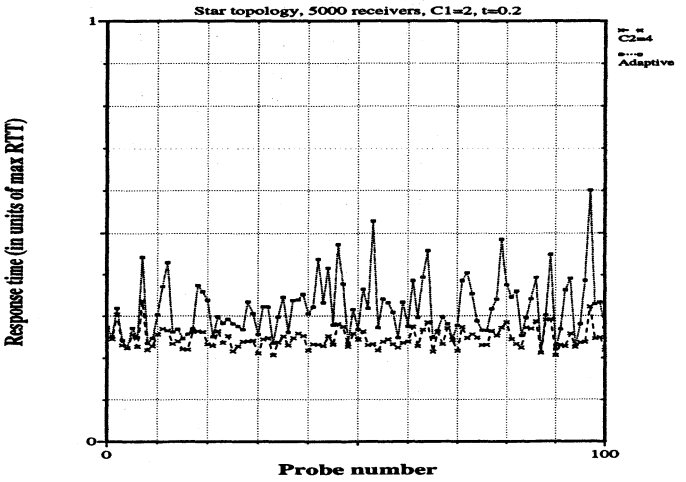
6.1 Adaptive feedback

In the previous section, it was shown that the performance of the proposed feedback mechanism needs some tuning to enhance its scalability for very large groups especially in the case when the worst state receivers are far from the sender, and most importantly far from each other. We focus on the worst state receivers because the outcome of the simulation experiments, discussed in the previous section, shows that almost all the excess replies that are generated in these cases are redundant worst case replies. This means that the shift in the start time of the timeout periods is still effective in eliminating replies from lower state receivers. Thus the parameter C_1 does not need tuning. It is the parameter C_2 which needs to be adapted to support very large groups. In other words, as the group size increases too much, the fixed value of $C_2 = 4$ no longer suffices to effectively suppress enough redundant replies. To this end we developed a simple adaptive algorithm that the sender uses to adapt the value of C_2 dynamically based on the number of received redundant replies. The sender counts the number of redundant worst state replies in response to a probe in the variable *dups*. Note that based on our previous results, the sender can safely count all replies coming in response to a probe assuming they are all worst state replies. Before sending a probe, the sender computes a new value for C_2 and appends it to the probe message. This value is used by the receivers in computing their random timeout periods. The algorithm which the sender applies is as follows.

AvgDups = α *AvgDups* + (1-α) *dups;*
If AvgDups > *THRESHOLD*
  C_2 = *Min*(C_2+1, *MAX_C2*);
Else



(a) number of replies



(b) response time

Figure 9 Effect of adaptive feedback

C_2 = *Max(*C_2*-1, MIN_C2);*

Figures 9(a) and (b) compare the performance of the static and adaptive feedback. In this simulation experiment, MIN_C2, MAX_C2, $THRESHOLD$, and α were set to 4, 50, 25, and 0 respectively. The figures show the ability of the simple adaptive algorithm to reduce the number of redundant replies drastically, without significant delay in response time. The tradeoff, however, is that it takes the sender a longer time before it can declare that the current epoch is over and no further replies will be received. Typically, the sender sends a new probe only at the end of an epoch, to avoid overlapping replies. The sender can always safely terminate an epoch after an amount of time equal to $(C_1 f(h) + C_2 g(h) + 2)\frac{RTT}{2}$ from sending a probe, where h is the highest state received in a reply to the current probe. After sending a probe, the sender sets a timer to expire after RTT plus the longest possible timeout period in the lowest state, for ending the epoch. As it receives replies, it adjusts this timer according to the above equation which is linearly proportional to C_2.

A more aggressive approach for ending an epoch without relying on C_2 would be to terminate the epoch after a period of time equal to RTT from the time of receiving the first reply. This aggressive approach safely assumes that any reply is coming from the highest state in the group. It attempts to give enough time for this reply to propagate to all other receivers and cause them to suppress their replies, if they haven't already sent it. The approach relies on the heuristic assumption that $RTT \cong \frac{RTT_{max}}{2}$.

If it is desired to limit the bandwidth taken by the reply packets to R, then the $THRESHOLD$ value can be set as a function of R. A simple approach is to set $THRESHOLD = \frac{R}{Reply\ size} \times Epoch\ duration$.

6.2 Passive feedback

The feedback mechanism, as described, keeps polling the receivers all the time. As soon as the sender determines that an epoch has ended, it immediately sends the next probe. While these probes are important for synchronizing the operation of the mechanism and avoiding potential spontaneous chains of status change notifications from receivers, yet in situations where the states of the receivers are stable for relatively long periods of time, this repeated probing is unnecessary.

One possible solution to optimize the performance of the feedback mechanism in such cases is to make the sender exploit the flexibility in spacing the probes, by increasing the idle time between ending an epoch and sending the following probe. However, this approach negatively affects the responsiveness of the feedback mechanism, especially when a change in state occurs after a relatively long stable state.

Another solution is to switch the feedback mechanism into *passive* mode whenever these relatively long stable states occur. When the sender gets similar state feedback from n consecutive probes, it sends a probe with a *passive flag* set, and carrying the current highest state h. Receivers do not respond to this probe, and the sender enters a passive non-probing mode. If a receiver detects that its state has risen above h, it immediately sets a timer in the usual way to report its state. On receiving a reported new higher state, each receiver updates the value of h. Similarly, if a highest state receiver detects that its state has fallen below h, it sets a timer in the usual way. However, when the receivers hear a report below h they do not update the value of h (as other receivers may be still in the h state). The sender, on receiving this report, switches back to the active probing mode, and the same cycle repeats.

7 CONCLUSION

In this paper, we presented a scalable and robust feedback mechanism for supporting adaptive multimedia multicast systems. Providing the source of a stream with feedback information about the used layers of the stream is crucial for the efficient utilization of the available resources. The feedback mechanism allows the sender to always send only layers for which interested receivers exist, and to suppress unused layers.

Simulation results showed that the proposed feedback mechanism scales well for groups of up to thousands of participants. For typical sessions with up to 100 participants (e.g., IRI sessions (Maly *et al.* 1997)), less than 10% of the receivers reply to a probe, in the worst case, while for larger sessions, of a few thousands of participants, the reply ratio is below 1.5%. The response time was found to be always below the maximum round-trip time from the sender to any of the group members.

The mechanism was shown to be robust in facing network losses, and to be more efficient than mechanisms which rely on session level messages for estimating individual round-trip times from each receiver to the sender. In addition, adaptive enhancements for supporting groups of up to 10,000 participants were proposed and shown to be effective in reducing the number of replies without a significant effect on response time.

Currently, we are incorporating the feedback mechanism in the *Quality of Session* control platform described in (Youssef *et al.* 1998), for further exploration and experimentation.

REFERENCES

Bolla, R., Marchese, M. and Zappatore, S.(1997) A Congestion Control Scheme for Multimedia Traffic in Packet Switching Best-Effort Networks. *Proceedings of the Second European Conference on Multime-*

dia Applications, Services and Techniques (ECMAST'97), Milan, Italy, May 1997.

Bolot, J., Turletti, T. and Wakeman, I.(1994) Scalable Feedback Control for Multicast Video Distribution in the Internet. *ACM SIGCOMM*, October 1994.

Bolot, J. and Turletti, T.(1994) A Rate Control Mechanism for Packet Video in the Internet. *Proceedings of IEEE INFOCOM'94*.

Busse, I., Deffner, B. and Schulzrinne, H.(1995) Dynamic QoS Control of Multimedia Applications based on RTP. *Second Workshop on Protocols for Multimedia Systems*, Salzburg, Austria, October 1995.

Cheung, S., Ammar, M. and Li, X.(1996) On the Use of Destination Set Grouping to Improve Fairness in Multicast Video Distribution. *Proceedings of IEEE INFOCOM'96*, San Francisco, CA, March 1996.

Deering, S. and Cheriton, D.(1990) Multicast Routing in Internetworks and Extended LANs. *ACM Trans. on Computer Systems*, **8(2), 85-110**, May 1990.

Floyd, S., Jacobson, V., Liu, C., McCanne, S. and Zhang, L.(1995) A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. *ACM SIGCOMM*, August 1995.

Gupta, A., Howe, W., Moran, M. and Nguyen, Q.(1995) Resource Sharing for Multi-Party Real-Time Communication. *Proceedings of IEEE INFOCOM'95*.

Jacobson, V.(1988) Congestion Avoidance and Control. *ACM Computer Communication Review*, **18(4), 314-329**, August 1988.

Li, X. and Ammar, M.(1996) Bandwidth Control for Replicated-Stream Multicast Video Distribution. *Proceedings of the Fifth IEEE Syposium on High Performance Distributed Computing (HPDC-5)*, Syracuse, NY, August 1996.

Maly, K., Abdel-Wahab, H., Overstreet, C.M., Wild, C., Gupta, A., Youssef, A., Stoica, E. and Al-Shaer, E.(1997) Interactive Distance Learning over Intranets. *IEEE Internet Computing*, **1(1), 60-71**, January 1997.

McCanne, S. and Jacobson, V.(1996) Receiver-driven Layered Multicast. *ACM SIGCOMM*, Stanford, CA, August 1996.

McCanne, S., Vetterli, M. and Jacobson, V.(1997) Low-complexity Video Coding for Receiver-driven Layered Multicast. *IEEE Journal on Selected Areas in Communications*, **16(6), 983-1001**, August 1997.

Senbel, S. and Abdel-Wahab, H.(1997) A Quadtree-based Image Encoding Scheme for Real-Time Communication. *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, Ottawa, Canada, June 1997.

Schulzrinne, H., Casner, S., Fredrick, R. and Jacobson, V.(1996) RTP: A Transport Protocol for Real-Time Applications. *RFC 1889*, January 1996.

Willebeek-LeMair, M. and Shae, Z.(1997) Videoconferencing over Packet-

Based Networks. *IEEE Journal on Selected Areas in Communication*, **15(6), 1101-1114**, August 1997.

Youssef, A., Abdel-Wahab, H. and Maly, K.(1998) The Software Architecture of a Distributed Quality of Session Control Layer. *Proceedings of the Seventh IEEE Syposium on High Performance Distributed Computing (HPDC-7)*, Chicago, IL, July 1998.

Youssef, A., Abdel-Wahab, H., Maly, K. and Gouda, M.(1997) Inter-Stream Adaptation for Collaborative Multimedia Applications. *Proceedings of the Second IEEE Symposium on Computers and Communications (ISCC'97)*, Alexandria, Egypt, July 1997.

Zhang, L, Deering, S., Estrin, D., Shenker, S. and Zappala, D.(1993) RSVP: A New Resource ReSerVation Protocol. *IEEE Network Magazine*, September 1993.

8 BIOGRAPHY

Alaa Youssef is a PhD candidate and research assistant in computer science at Old Dominion University. His research interests include networking support for multimedia systems, resource management in heterogeneous distributed systems, and multimedia collaborative and tele-teaching systems. Youssef received his BSc and MSc in computer science from Alexandria University, Egypt, in 1991 and 1994, respectively.

Hussein Abdel-Wahab is a professor of computer science at Old Dominion University, an adjunct professor of computer science at the University of North Carolina at Chapel Hill, and a faculty member at the Information Technology Lab of the National Institute of Standards and Technology. His main research interests are collaborative desktop multimedia conferencing systems and real-time distributed information sharing. Abdel-Wahab received a PhD in computer communications from the University of Waterloo. He is a senior member of IEEE Computer Society and a member of the ACM.

Kurt Maly is a Kaufman professor and chair of computer science at Old Dominion University. His research interests include modeling and simulation, very high performance network protocols, reliability, interactive multimedia remote instruction, Internet resource access, and software maintenance. Maly received a PhD in computer science from the Courant Institute of Mathematical Sciences, New York University. He is a member of the IEEE Computer Society and the ACM.