Architectural Issues for Multicast Congestion Control

Anindya Basu
Bell Laboratories
basu@research.bell-labs.com

S. Jamaloddin Golestani
Bell Laboratories
jamal@research.bell-labs.com

Abstract

As multicast applications are becoming more common, multicast congestion control has become an important problem. In this paper, we present an architecture for multicast congestion control schemes. We identify three components that constitute this architecture: namely, the feedback consolidation mechanism, the round-trip time measurement mechanism, and the loss detection and timeout mechanism. We then show how these mechanisms can be implemented for both rate-based and window-based congestion control schemes under various feedback topologies. Finally, we discuss the tradeoffs involved in these implementations when we use the different congestion control schemes (rate-based vs. window-based) and feedback topologies.

1 Introduction

Multicast congestion control has become an important problem as multicast applications are becoming more widespread on the Internet. A large class of multicast applications use reliable multicast which requires lossless data delivery from a sender to a receiver group. For example, a software vendor could use this service to distribute the latest version of its software to its registered customer base[1]. Over the last year, multiple congestion control schemes for reliable multicast have been proposed [RVC98, GS99, Chi98, WC98, HF98]. These schemes come in a variety of flavors: for example, a congestion control scheme could be rate-based or window-based depending on whether the transmission rate or the the window size is adjusted in response to network congestion. At the same time, a congestion control scheme could be tree-based [PSLB97] or broadcast-based [FJM+95] depending on the topology used to send feedback to the sender. It is thus clear that the design space for such schemes is extremely diverse and careful analysis is required in order to identify the relative merits and demerits of a specific scheme.

A multicast congestion control scheme has two main components: a policy component that enforces a specific congestion control discipline, and a monitor component that monitors the network and provides information about the state of congestion in the network to the policy component. It is the policy component that determines, for example, whether the regulation parameter [GS99] is the transmission rate or the window size, and whether the feedback topology is tree-based or broadcast-based. The monitor component supports the policy component by collecting and consolidating information about the network state, such as round trip times and packet losses and makes it available to the policy component. This information is used by the policy component to regulate the transmission rate or the window size. In short, the monitor component provides a basic architecture that can be used to build specific congestion control schemes, as determined by the policy component. In this paper, we focus on this architecture. We first identify what network related information a monitor component needs to provide to a policy component. We then study how this architecture (or the method of collecting this information) is affected by the choice of different parameters in the policy component.

The main contribution of this paper is a detailed study

[1] This is in contrast to multicast applications that do not require every data packet to be delivered at the destinations. For example, applications that use multicast to distribute audio and/or video over the Multicast Backbone (MBONE) can tolerate dropped packets up to some threshold limit.

of alternative architectures for constructing congestion control schemes for reliable multicast. We identify three basic primitives (or mechanisms) that constitute this architecture: the feedback consolidation mechanism, the round trip time measurement mechanism, and the loss detection and timeout mechanism. We then present how these mechanisms can be implemented in a variety of scenarios that are determined by the configuration of the policy component. For example, we try to answer questions such as "What effect does the choice of rate or window size as a regulation parameter have on how we consolidate feedback?" or "How should we estimate round trip times when a tree-based feedback mechanism is used as opposed to when a broadcast-based feedback mechanism is used?" and so on. The purpose of this paper is not to study specific congestion control schemes or to propose new ones, but rather to develop a framework for studying general techniques for doing multicast congestion control. This will enable us to understand the tradeoffs involved when choosing a certain congestion control scheme for a specific application.

We find that if round-trip times are to be estimated in a scalable manner, it is necessary that the receivers be organized into some kind of hierarchy to send feedback. The more well-defined the hierarchy, the more accurate the round-trip time estimates. We also find that in the absence of a well-defined hierarchy, rate-based schemes allow for more scalable feedback consolidation (and therefore reduced feedback overheads) than window-based schemes. Finally, we find that timeouts can be detected more accurately when a well-defined hierarchy exists for the receivers.

The rest of the paper is organized as follows: Section 2 provides a brief overview of the requirements for multicast congestion control and identifies the three mechanisms that constitute the monitor component. Section 3 explores how feedback consolidation can be scalably done under a variety of environments. Section 4 discusses how round-trip times can be estimated in a scalable manner. Section 5 describes mechanisms for loss detection and timeouts. We discuss the implications of our findings and conclude in Section 6.

2 Requirements for Reliable Multicast Congestion Control

An end-to-end congestion control scheme controls network congestion by regulating traffic at the sender. Two well-know types of end-to-end traffic regulation are the rate-based and the window-based schemes. In rate-based schemes, the *regulation parameter* is the transmission rate (or the sending rate) r_s. This means that traffic is regulated by keeping r_s at or below an acceptable level. Rate-based schemes work in the same way for both unicast and multicast communication. In window-based schemes, the regulation parameter is the number of outstanding packets to the receiver(s) and the traffic is controlled by keeping this number at or below the assigned window size. It is shown in [GS99] that in multicast communication, window-based regulation requires one window size w_j per receiver j: the number of outstanding packets to each receiver must be independently controlled and kept below the corresponding window size. Applying a common window size to all receivers typically results in restricting the multicast transmission rate beyond the requirements of congestion control. Therefore, window-based schemes for multicast must use N regulation parameters w_j, where N is the number of receivers.

An important part of traffic regulation is to adaptively determine the regulation parameter(s) of choice, based on the network congestion status. In rate-based regulation, the approach that has been commonly proposed for determining the rate r_s, is to first determine an acceptable rate r_j for each receiver j, solely based on congestion status of the network path leading to j, and regardless of other receivers' status. The actual multicast transmission rate r_s is then set to

$$r_s = \min_j r_j \ . \tag{1}$$

This approach for determining the multicast transmission rate may be called *unicast decomposition* since the nominal receiver rates r_j are determined based only on the status of the communication path to j. In window-base regulation, the unicast decomposition approach translates to determining each window size w_j, solely based on the congestion status of the path leading to j [GS99]. In general, one can conceive of other ways of updating the regulation parameters. However, our dis-

cussion in this paper is based on the unicast decomposition approach, since it leads to simpler implementations.

The current Internet lacks mechanisms for explicit congestion notification. Thus, the only available signs of congestion are the packet losses and packet delays that are observable by each receiver. Therefore, an algorithm for updating a receiver's nominal rate r_j or window size w_j must make use of the loss and delay (or round trip time) measurements pertaining to packets traveling towards that receiver. Unlike TCP congestion control where such an update algorithm is run at the sender, scalable implementations of congestion control for multicast communication require a *receiver-driven* approach. In this approach, each receiver j is responsible for updating its rate r_j or window size w_j based on its delay and loss observations. As (1) indicates, in rate-based regulation, the receiver should provide a feedback with the updated value r_j to the sender (or some other entity in the group) so that the transmission rate r_s may be determined. In window-based regulation, however, the updated window size w_j is not the right feedback to be sent by j. As shown in [GS99], each receiver should compute a number n_j as the maximum packet sequence number that it expects and send a feedback with that value. This parameter n_j is a function of the window size w_j and other information locally available at j [GS99]. The sender performs traffic regulation by only sending packets up to the sequence number n_{send}, determined by

$$n_{\text{send}} = \min_j n_j \ . \tag{2}$$

We refer to the ensemble of activities needed to execute (1) or (2), as *consolidation of receiver feedback*.

We now summarize the results of the above discussion. Implementing an end-to-end, receiver-driven, multicast congestion control strategy, based on unicast decomposition, requires mechanisms for executing the following activities in a scalable manner.

– A mechanism for consolidating receiver feedback in (1) or (2).

– A mechanism for estimating receiver round trip times, and

– A mechanism for detecting packet losses and timeouts.

In this paper, we describe an architecture composed of the above three elements to support implementation of congestion control. We are not interested in the form and details of algorithms needed to update receiver rates or window sizes, beyond asserting that such algorithms utilize the message loss, and round trip time (RTT) information available to each receiver for the following purposes:

– The loss information serves as the main indication of congestion, in the current Internet where no mechanism for explicit congestion notification exists.

– The round trip time estimation, if done on a short time scale, provides an additional measure of network congestion.

– More importantly, the receiver round trip time information could be necessary in order to achieve a certain fairness criterion in congestion control. As shown in [GS99], if the fairness criterion is such that the average throughput is inversely proportional to the round-trip time (a la TCP), then rate based regulation scheme require the estimation of round trip times. On the other hand, if the fairness criterion is such that the average throughput is independent of the round-trip time, then window based regulation schemes require estimation of round trip times. Thus, in order to achieve a given fairness criterion in congestion control schemes, it is not possible to implement both rate-based and window-based schemes without estimating round-trip times. For at least one of these two forms of regulation, receiver RTTs must be known.

A particularly important case in loss detection is the detection of back-to-back losses up to (and including) the most recent message from the sender. It is important that the receivers be able to distinguish this case from the situation where the sender simply stops sending. Timeouts are necessary for this purpose and thus the appropriate setting of timeouts is an important function.

We now briefly describe the elements in the policy component that affect the architectural elements. For the purposes of this paper, we have identified two such elements — the regulation parameter and the feedback topology. The regulation parameter can either be the transmission rate, in which case the conges-

tion control mechanism is rate-based, or the window sizes, in which case the congestion control mechanism is window-based. For example, TCP uses a window-based mechanism whereas the proposed TCP Friendly Reliable Multicast Congestion Control Algorithm (TFMCC) [HF98] is a rate-based one. As we shall see in Section 3, the choice of the regulation parameter has important consequences for the architecture. The second policy element that affects the architecture is the feedback topology. A common feedback topology is the tree-based feedback topology where the receivers are organized into a tree (typically at transport layer) with the sender at the root, and each receiver sends its feedback to its parent in the tree. The RMTP protocol [PSLB97] uses this mechanism. Alternatively, a broadcast-based feedback topology could be used, such as in the SRM protocol [FJM$^+$95]. In this case, each receiver broadcasts its feedback to the entire multicast group (or a subset thereof). Excess feedbacks are suppressed using randomization or some other technique, so as to avoid the feedback implosion problem. Note that the feedback topology is different from the feedback consolidation mechanism — they are undoubtedly tightly coupled, but the feedback topology specifies who to send the feedback to and the feedback consolidation mechanism specifies how to make the feedback sending mechanism scale. In the next few sections, we discuss each of the architectural elements in greater detail and show how they are affected by the different choices for the regulation parameter and the feedback topology.

3 Consolidation of Receiver Congestion Control Feedback

While the feedback consolidation in (1) or (2) is conceptually simple, it still poses a scalability problem at the sender if the sender is left with the task of receiving and processing feedback from all receivers and performing the minimization. This scalability problem may be alleviated in a number of ways. One solution would be to organize the receivers into a hierarchy of groups and subgroups. The consolidation can now be performed in multiple stages by first consolidating the feedback of receivers within the smallest subgroups and then progressively consolidating feedback in larger subgroups until it is complete [GS99]. This solution can be implemented using a tree-based topology either at the transport layer or at the network layer. Since the current Internet does not have any network layer support for congestion control, we use a transport layer tree topology for sending feedback. Another solution would be to suppress feedback from receivers that have been determined not to have the smallest feedback. One way to accomplish this is with a broadcast-based feedback topology where every receiver can hear the feedback sent by other receivers. A receiver that hears a feedback smaller than its own suppresses its own feedback. This approach is similar to the technique used for error control in the SRM protocol [FJM$^+$95]. Hybrid topologies based on various combinations of the tree- and broadcast-based topologies can also be used. In the rest of this section, we study feedback consolidation using these topologies for both rate-based and window-based congestion control. However, before considering any specific feedback topology, we describe a feedback suppression technique that can be used in any feedback topology.

3.1 Suppression of Receiver Feedback

We consider rate-based congestion control first. We assume that congestion control feedback r_j is sent from each receiver j directly to the sender. We also assume that the current transmission rate is known to the receivers. This information can be explicitly included in the packet headers at the time of transmission. Furthermore, the receivers may send feedback either periodically, or when certain special packets (typically flagged by the sender) are received. In either case, we refer to the interval between successive rounds of feedback as the *feedback interval*.

Let r_s^{past} be the most recent transmission rate of the sender. Let the smallest feedback received by the sender during the previous feedback interval be from receiver b, i.e., let $r_b^{\text{past}} = r_s^{\text{past}}$. Denote by $\delta_{\max}$ the maximum increase that the rate r_b of receiver b could have undergone during the past feedback interval. Based on the algorithm used to update receiver rates, an explicit expression can be provided for $\delta_{\max}$ as a function of r_s^{past} and the duration of the most recent feedback interval. It follows that the feedback r_b that receiver b now provides to the sender cannot be more than $r_b^{\text{past}} + \delta_{\max}$. Therefore, unless receiver b has dropped out of the multicast group, the transmission rate, r_s, of the sender after the

next update will satisfy:

$$r_s \leq r_s^{\text{past}} + \delta_{\max}. \quad (3)$$

It follows that a receiver j with the updated rate r_j can suppress its feedback if

$$r_j > r_s^{\text{past}} + \delta_{\max}, \quad (4)$$

because the new value of r_j will have no effect in updating the value of r_s.

We now ask the following question: if the feedback value of a receiver remains (almost) unchanged after a feedback interval, should the receiver send the feedback? The answer depends on whether the sender, after each feedback interval, removes the feedback value that it has received from each receiver, or stores it and continues to apply it in subsequent updates until it receives a new feedback from the same receiver. We refer to these two scenarios as *memoryless* and *with-memory* consolidation, respectively. It may appear that with-memory consolidation allows for better suppression since a feedback need not be sent unless its value has changed. This is not always true: let r_j^{last} denote the last feedback value sent by j to the sender. Assume that, after an update, the new feedback value r_j satisfies the condition stated in (4). In memoryless consolidation, the feedback can be safely suppressed by j. However, in with-memory consolidation, the suppression would be allowed only if the outdated feedback value r_j^{past}, which is kept at the sender, also satisfies

$$r_j^{\text{last}} > r_s^{\text{past}} + \delta_{\max}. \quad (5)$$

Otherwise, r_j^{last} could become the smallest feedback value at the sender during the next update and restrict the new sender rate, unnecessarily. Note that r_j^{last} and r_s^{past} do not necessarily belong to the same feedback interval; while r_s^{past} is the transmission rate during the most recent interval, r_j^{last} is the most recent feedback sent by j, which may have been sent during any previous interval.

The above observations for the cases of memoryless and with-memory consolidation may be summarized as follows:

Memoryless Consolidation: Feedback may be suppressed by a receiver j, provided that r_j satisfies (4).

With-memory Consolidation: Feedback may be suppressed by a receiver j, provided that one of the following conditions are met:

$$\min(r_j, r_j^{\text{last}}) > r_s^{\text{past}} + \delta_{\max}, \quad (6)$$

or,

$$r_j \approx r_j^{\text{last}}. \quad (7)$$

Qualitatively speaking, when receiver rates typically undergo small changes during an update, with-memory consolidation results in better suppression. Conversely, if the rate update algorithm is more dynamic and results in larger changes in receiver rates, memoryless consolidation could allow for more suppression, since it is less likely that (6) would be satisfied even when (4) is met.

In general, $\delta_{\max}$ should be a relatively small value since the increments in receiver rates during rate updates are typically small. Therefore, the above suppression policy in both memoryless and with-memory cases leads to substantial reduction in the number of feedback messages sent to the sender. Typically, a small fraction of receivers would need to send feedback during each feedback interval. Nevertheless, the scalability improvement provided by the above suppression policy is relative; with a sufficiently large multicast group, the sender could still suffer from the feedback implosion problem. In Sections 3.3 and 3.2, we shall discuss other feedback consolidation techniques that either improve the performance of the feedback suppression mechanism or make the consolidation mechanism scale better.

We consider window-based congestion control next. For a receiver j, let w_j and m_j denote the window size of j and the highest packet sequence number received by j. Assuming that all packets with sequence numbers up to m_j have been received, the feedback parameter n_j is calculated by j as [GS99],

$$n_j = m_j + w_j. \quad (8)$$

We now determine the conditions under which receiver j can suppress its feedback. Let $m_j + o_j$ be the highest sequence number of a packet that has been sent by the sender at the time it receives a feedback (if not suppressed) from j for packet n_j. It follows that o_j is the number of outstanding packets that have been sent to receiver j when feedback for n_j arrives at the sender. We can further say that the feedback for n_j can be suppressed if n_j is sufficiently larger than $m_j + o_j$. Alternatively, using (8), we can say that the feedback for n_j can be suppressed if w_j is sufficiently larger than o_j.

There are two major problems in determining o_j. First, the receiver needs to determine the number of outstanding packets at a time in the future: i.e., the time when the feedback for n_j would reach the sender (if not suppressed). This issue can be addressed by using some upper bound on o_j, based on the number of outstanding packets to j at some recent time in the past. For example, let o_j^{past} denote the number of outstanding packets to j when packet m_j was sent by the sender. It may be possible to find a value $\gamma_{\max}$ such that the following upper bound would hold:

$$o_j \leq o_j^{\text{past}} + \gamma_{\max}. \tag{9}$$

Using this result, the decision to suppress the feedback for n_j would depend on whether or not w_j is larger than $o_j^{\text{past}} + \gamma_{\max}$. The second problem is that in large multicast groups, determining the number of outstanding packets to j, even at some time in the past, is not feasible. This would involve sending acknowledgments from each receiver j to the sender for some or all packets, having the sender determine the number of outstanding packets for each j, and reporting it back to j. This procedure does not scale and produces additional feedback messages instead of suppressing them.

In conclusion, while the transmission rate can be readily determined by each receiver in a multicast group, the same is not true about the number of outstanding packets to the receivers. Notice also that the transmission rate is the same for the whole multicast group, while the number of outstanding packets vary from receiver to receiver. Because of this fundamental difference between determining the rate and the number of outstanding packets, we conclude that the feedback suppression method discussed above for rate-based congestion control cannot be extended to window-based congestion control without severely limiting scalability.

3.2 Feedback Consolidation for Broadcast-based Topologies

We now consider a pure broadcast feedback topology in which each receiver broadcasts its feedback messages to the entire multicast group. In [HF98], a feedback suppression technique for rate-based congestion control has been described which exploits the fact that each receiver hears the feedback sent by other receivers. We refer to it as the *random wait* technique. The idea is very similar to the one used in [FJM+95] for the suppression of error control messages. Once the sender polls the receivers to send their congestion control feedback, each receiver schedules the transmission of its feedback after a random delay. However, the feedback is suppressed if, prior to the scheduled time, the receiver hears a feedback from some other receiver with a lesser or equal value. The delay distribution used to schedule the sending of the feedback depends on the feedback value to be reported, such that receivers with a smaller feedback value (and thus more likely to form the bottleneck) tend to send their feedback sooner. A random (rather than a fixed) delay is used so that the transmission of feedback by receivers with almost equal feedback parameters would not coincide. This would allow for the suppression of similar feedback messages which have been scheduled for a later time.

The performance of the above feedback suppression scheme is still to be studied. However, it is qualitatively clear that the amount of suppression that can be achieved by this technique depends on the ratio of the maximum permissible delay in scheduling feedback to receiver round trip times. Unless the feedback from different receivers can be spread over several round trip times, the achievable suppression would be negligible.

The random wait suppression scheme can be be combined with the suppression technique proposed in Section 3.1 to achieve better performance. In this combined scheme, a feedback will not be scheduled if the feedback value exceeds the threshold specified in (4) (or, in the case of with-memory consolidation, satisfies (6) or (7)).

Finally, we note that the random wait suppression scheme, like the technique of Section 3.1, cannot be extended to window-based congestion control. The essence of a window-based scheme is to provide quick feedback from receivers in order to achieve tight control at the sender over the number of outstanding packets to each receiver. This approach is inconsistent with delaying receiver feedback for up to several round trip times, which is necessary to enable sufficient suppression.

3.3 Feedback Consolidation for Tree-based Topologies

In sections 3.1 and 3.2, we attempted to improve scalability by suppressing feedback messages that could be avoided. In this section, we summarize an alternative ap-

proach for improving scalability [GS99]. We consider a multicast session with receivers organized hierarchically in a tree, with the sender at the root, as shown in Figure 4.1. In order to eliminate feedback implosion at the sender, the consolidation of feedback messages can be performed in a distributed fashion at each node in the tree. Each receiver in the tree periodically sends congestion control feedback only to its parent in the tree. The parent consolidates the feedback information from all its children along with its own and sends it up the tree. Note that each receiver can either send its feedback to its parent as soon as the feedback value changes, or it can send this value periodically (such as every δ_j units of time).

Feedback consolidation using a tree-based topology scales well. For any group size, a hierarchical group organization always exists such that the number of children of the sender or any parent receiver could be limited as desired. Moreover, this approach is equally applicable to rate-based and window-based congestion control.

Finally, we note that in rate-based congestion control, the above tree-based consolidation scheme can, in principle, be combined with the feedback suppression technique of Section 3.1 to further minimize the number of feedback messages processed by each entity. However, the improvement obtained from this combination is negligible except for tree-based topologies where the number of children per parent is very large.

4 Estimation of Round-Trip Times

In this section, we describe algorithms for measuring round-trip times in a multicast setting. For this purpose, we first define what we mean by the notion of a round-trip time in a multicast environment. Typically, the round-trip time between a sender and a receiver is equal to the time elapsed between the sender sending a packet and receiving an acknowledgment for it from the receiver. In unicast communication, the packet round trip times are measured by taking the difference between the time a packet is sent out and the arrival of its acknowledgment at the sender. Since both the send time and the time of acknowledgment arrival are measured at the sender, clock synchronization between the sender and the destination is not required.

This approach cannot be used in multicast communication since it would lead to the *ack implosion* problem at the sender for large multicast groups. One possible solution would be to measure one-way delays at the receivers assuming some form of global clock synchronization between the sender and the receivers. However, this approach has a scalability problem since clock synchronization algorithms depend on all-to-all broadcast messages. An alternative approach would be to have all members of the multicast group synchronize to GPS receivers, but such systems are still not commonplace. To address this issue, we present a family of round-trip time estimation algorithms that do not require clock synchronization, are scalable, and work for multiple feedback topologies.

4.1 The Tree-based Feedback Topology

We first consider a multicast session with a tree-based feedback topology (Figure 4.1) in which each receiver j regularly sends congestion control feedback information to its parent. We define the round trip time for a receiver j to be equal to the time that elapsed between the sender multicasting a packet p (to all receivers in the group including j) and the potential impact of j's acknowledgment for p reaching the sender. We use the term "potential impact of j's acknowledgment for p reaching the sender" since an acknowledgment from a receiver low down in the feedback tree never directly reaches the sender, but is filtered and aggregated by intermediate nodes. Now, the average round trip time τ_j for receiver j can be expressed as,

$$\tau_j = \tau_{s,j} + \delta_j + \tau_{j,s} \,, \tag{10}$$

where $\tau_{s,j}$ denotes the average one-way delay associated with the forward path from the sender to receiver j, $\tau_{j,s}$ denotes the average time between the transmission of a feedback message by receiver j and the potential impact of this message reaching the sender, and δ_j is the average time between the arrival of a packet at j and the transmission of the first congestion control feedback out of j. For example, if congestion control feedback messages are sent by receiver j periodically, with a period of Δ_j, the time between the arrival of a packet at j and the transmission of the first congestion control feedback out of j is uniformly distributed between 0 and Δ_j. This implies that $\delta_j = \Delta_j/2$.

Denoting the parent of receiver j in the tree hierarchy

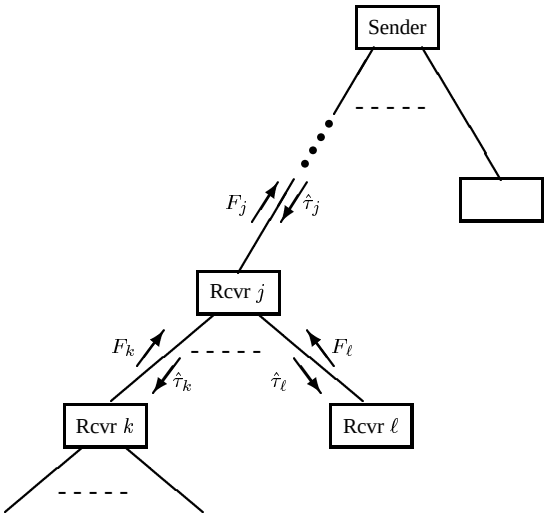

Figure 1: A hierarchical organization of receivers for upward consolidation of congestion control feedback and downward distribution of delay information.

as j', the delay term $\tau_{j,s}$ in (10) can be expressed as

$$\tau_{j,s} = \tau_{j,j'} + \delta_{j'} + \tau_{j',s} \tag{11}$$

where $\tau_{j,j'}$ denotes the average one-way delay for the congestion feedback from receiver j to reach its parent j'. Thus $\tau_{j,s}$ is the sum of the average time for the feedback to go from j to j' and then from j' to the sender s. The $\delta_{j'}$ term accounts for the average time between the arrival of a feedback message from the child j and the transmission of the first feedback message out of j'.

We now develop a recursive algorithm in which the round trip time τ_j of each receiver j is determined using the round trip time $\tau_{j'}$ of its parent j'. To this end, we first use an expansion similar to (10) for $\tau_{j'}$,

$$\tau_{j'} = \tau_{s,j'} + \delta_{j'} + \tau_{j',s} \ . \tag{12}$$

Combining (10), (11), and (12) we get,

$$\tau_j - \tau_{j'} = \tau_{s,j} + \delta_j + \tau_{j,j'} - \tau_{s,j'} \tag{13}$$

We now show that the delay difference $\tau_j - \tau_{j'}$ in equation 13 can be locally determined by the parent j'. Consider a multicast packet p. Let t_s^p denote the time at which p is transmitted by the sender. Let t_j^p, $t_{j'}^p$ denote the arrival time of p at j, j', respectively. We denote by $\text{fdbk}(p,j)$ the first congestion control feedback sent from j to j' after time t_j^p. Let this feedback arrive at j' at time $t_{j'}^{\text{fdbk}(p,j)}$. Clearly,

$$E\{t_{j'}^p\} = t_s^p + \tau_{s,j'}, \tag{14}$$

and

$$E\{t_{j'}^{\text{fdbk}(p,j)}\} = t_s^p + \tau_{s,j} + \delta_j + \tau_{j,j'} \ . \tag{15}$$

Now, consider the quantity

$$\theta_{j,j'}^p \stackrel{\Delta}{=} t_{j'}^{\text{fdbk}(p,j)} - t_{j'}^p. \tag{16}$$

This quantity is the difference in time between the occurrence of two events: the first event occurs when j' receives the packet p and the second, when j' receives the first feedback message that j sent out after j received p. Since these two events are both observed by the same receiver j', their time difference can be measured without any clock synchronization. Furthermore, by combining (13) – (16), we may express the average value of $\theta_{j,j'}^p$, as follows

$$\begin{aligned} \theta_{j,j'} &\stackrel{\Delta}{=} E\{\theta_{j,j'}^p\} = E\{t_{j'}^{\text{fdbk}(p,j)\}} - E\{t_{j'}^p\} \\ &= \tau_{s,j} + \delta_j + \tau_{j,j'} - \tau_{s,j'} \\ &= \tau_j - \tau_{j'} \ . \end{aligned} \tag{17}$$

Or, equivalently,

$$\tau_j = \theta_{j,j'} + \tau_{j'} \ . \tag{18}$$

In summary, we have shown that the round trip time of each receiver j can be recursively determined by adding $\theta_{j,j'}$ to the round trip time of its parent. The quantity $\theta_{j,j'}$, for each receiver j, may be determined by its parent j'. This can be done by measuring $\theta_{j,j'}^p$ for different packets p and determining their average. We refer to $\theta_{j,j'}$ as the *RTT differential* of receiver j.

Note that for the parent j' to measure $\theta_{j,j'}^p$, it must be able to identify which of the feedback messages received from j is the first one sent following the arrival of p at j. In order to facilitate this discussion, we now make a distinction between the congestion control feedback and per-packet acknowledgments that are sent by each receiver j to its parent (for all or some of the packets received by j) in order to assist the parent in calculating the RTT differential of j. The distinction made here between the congestion control feedback and the acknowledgments is an abstract one; in reality both types of information could be embedded in the same physical message. Using the notation introduced earlier, let t_j^p denote the time when p arrives at j and let $t_j^{Ack(p,j)}$ denote the time when j sends out the acknowledgment for p. Furthermore, let $t_{j'}^{Ack(p,j)}$ denote the time when

j' receives this acknowledgment, and let ϕ_j denote the expected time it takes j to send the acknowledgment for packet p after it has received p, i.e.,

$$\phi_j \stackrel{\Delta}{=} E\{t_j^{Ack(p,j)} - t_j^p\} \; . \tag{19}$$

Finally, let $\hat{\theta}_{j,j'}$ denote the expected time difference between the arrival of p at j' and the arrival of the corresponding acknowledgment from j, i.e.,

$$\hat{\theta}_{j,j'} \stackrel{\Delta}{=} E\{t_{j'}^{Ack(p,j)} - t_{j'}^p\}. \tag{20}$$

We call this the *adjusted RTT differential*. It follows that

$$\begin{aligned} \hat{\theta}_{j,j'} &= t_s^p + \tau_{s,j} + \phi_j + \tau_{j,j'} - (t_s^p + \tau_{s,j'}) \\ &= \theta_{j,j'} - \delta_j + \phi_j \; , \end{aligned} \tag{21}$$

where the last equality results from (13). The difference between the quantity $\hat{\theta}_{j,j'}$ which can be measured by j' and the desired quantity $\theta_{j,j'}$ consists of two terms ϕ_j and $-\delta_j$, both of which can be measured by the receiver j.

Now suppose that j' knows its round trip time $\tau_{j'}$, and the adjusted RTT differential $\hat{\theta}_{j,j'}$ for its child j. Then j' computes the parameter

$$\hat{\tau}_j \stackrel{\Delta}{=} \tau_{j'} + \hat{\theta}_{j,j'} \tag{22}$$

and sends it down to j. Using (18), (21) and (19), j can determine its round trip time τ_j as

$$\tau_j = \hat{\tau}_j + \delta_j - \phi_j \; . \tag{23}$$

Thus, we conclude that the receiver round trip times in a multicast group organized as a tree hierarchy can be estimated using the following algorithm:

- Each receiver j that is a parent in the tree hierarchy maintains an estimate of the adjusted RTT differential $\hat{\theta}_{k,j}$ for each child k. Each time j receives a packet p from the sender and an acknowledgment for p from k, it uses the time difference between these two events to update $\hat{\theta}_{k,j}$. The update may be done by an exponentially weighted algorithm or some other averaging process. This calculation is also performed by the sender for each of its children, i.e., for each receiver which is just one level below the sender in the tree hierarchy.

- Each receiver j maintains an estimate of the parameter ϕ_j. Every time it sends an acknowledgment for a packet p to its parent, it uses the time elapsed between the arrival of p and the sending of the acknowledgment for p to update ϕ_j.

- Each time a receiver j receives a new feedback from j' about its adjusted round trip time $\hat{\tau}_j$, it updates its round trip time as,

  $$\tau_j = \hat{\tau}_j + \delta_j - \phi_j. \tag{24}$$

  It also updates the adjusted round trip time of each child k as

  $$\hat{\tau}_k = \tau_j + \hat{\theta}_{k,j} \tag{25}$$

  and sends a message with the updated information to k. This process is initiated every T seconds by the sender, which sends to each child k its adjusted round trip time $\hat{\tau}_k = \hat{\theta}_{k,s}$. (Recall that for the sender s, $\tau_s = 0$.)

A more detailed description of this algorithm and some applications can be found in [BG98]. A similar algorithm was independently developed and presented at the RMRG meeting in Dec, 1998 [Rhe98].

4.2 The Broadcast-based Feedback Topology

We now come to the case of the broadcast-based feedback topology. As explained in Section 3.2, in its purest form, a broadcast-based feedback topology works as follows: on the receipt of a packet, a receiver waits for a period of random duration to see if it receives a feedback from some other receiver that is less than or equal to its own feedback. If not, it broadcasts its own feedback to the multicast group. In such a scenario, it is very difficult, if not impossible, to estimate RTTs at the receivers because a particular receiver may never send any feedbacks. We therefore solve the problem for a slightly more structured topology, namely, the hop-scoped feedback topology that is described in the next section.

4.3 The Hop-Scoped Feedback Topology

The idea of a hop-scoped feedback topology has been previously used for error-control [LESZ98]. In this topology, each receiver sends out a feedback with a time-to-live (TTL) field that limits the scope to all neighbors

within a certain number of hops. Any neighbor who receives such a feedback, consolidates it with all the other feedbacks it has received, and rebroadcasts it (using the same hop scoped mechanism) till it reaches the sender. To solve the RTT measurement problem in this topology, we first solve the problem in an intermediate, multi-parent hierarchical topology, and then show how to extend the solution to the hop-scoped topology.

The Multi-Parent Feedback Topology: In the multi-parent hierarchical feedback topology, the receivers are organized as in the tree-based case, except that each node can now have multiple parents. The only exceptions are the direct descendants of the root, which have a single parent.

Each receiver in the multi-parent tree topology sends its congestion control feedback to all its parents. Each parent then consolidates feedbacks from all its children and sends the consolidated feedback up the tree till the feedback reaches the root. This is possible because the multi-parent hierarchical topology is *loop-free* i.e., it is not possible to start from a receiver j and follow some child-parent chain in the topology to end up at j once more.

It is clear from the above description that the potential impact of a receiver's congestion control feedback can reach the sender through multiple paths. Therefore, the round trip time is determined by the feedback path with the smallest delay. Note that the feedback path with the minimum delay may change on a per-packet basis.

We shall now show that the iterative approach (described in the last section) for estimating round trip times is in essence applicable to the multi-parent hierarchical topology. This approach uses the delay differential measured at each receiver. However, since we need to determine for each packet, the smallest round trip time over multiple feedback paths, the estimation of round trip times requires some additional steps.

Consider a receiver j and let $\mathcal{P}_j$ denote the set of parents of j, i.e. the receivers to which j sends its congestion control feedback. Let $\text{fdbk}(m,j)$ denote the first congestion control feedback sent by j, following the arrival of message m at j. The message m can either be a feedback message sent by a child of j, or a data packet sent by the sender. Denote by $\delta(m,j)$ the time elapsed between the arrival of m at j and the transmission of $\text{fdbk}(m,j)$ by j. Consider a parent $j' \in \mathcal{P}_j$, and a given packet p. Denote by $\tau_j^{p,j'}$ the round trip time of packet p for receiver j, assuming that j has no parent other than j'. Then,

$$\tau_j^{p,j'} \triangleq t_s^{\text{fdbk}(p,j),j'} - t_s^p \,, \tag{26}$$

where t_s^p is defined in Section 4.1 and $t_s^{\text{fdbk}(p,j),j'}$ denotes the time at which the potential impact of $\text{fdbk}(m,j)$ reaches the sender, assuming that j has no parent other than j'. When j has multiple parents, the earliest time at which the potential impact of $\text{fdbk}(m,j)$ reaches the sender is

$$t_s^{\text{fdbk}(p,j)} \triangleq \min_{j' \in \mathcal{P}_j} t_s^{\text{fdbk}(p,j),j'} \,. \tag{27}$$

Therefore, τ_j^p, or the round trip time of p to receiver j, would be

$$\begin{aligned} \tau_j^p &\triangleq t_s^{\text{fdbk}(p,j)} - t_s^p \\ &= \min_{j' \in \mathcal{P}_j} \tau_j^{p,j'} \,. \end{aligned} \tag{28}$$

Let $\tau_{k,\ell}^m$ denote the time taken by message m to go from k to ℓ, where m represents a data packet or feedback message, and k, ℓ represent nodes in the feedback hierarchy. Using this notation, we can expand $\tau_j^{p,j'}$ as

$$\begin{aligned} \tau_j^{p,j'} &= \tau_{s,j}^p + \delta(p,j) + \tau_{j,s}^{\text{fdbk}(p,j)} \\ &= \tau_{s,j}^p + \delta(p,j) + \tau_{j,j'}^{\text{fdbk}(p,j)} + \\ &\quad \delta(\text{fdbk}(p,j),j') + \tau_{j',s}^{\text{fdbk}(\text{fdbk}(p,j),j')} \,. \end{aligned} \tag{29}$$

Next, notice that the quantity $\theta_{j,j'}^p$, earlier defined in (16), may be expressed as

$$\begin{aligned} \theta_{j,j'}^p &= t_{j'}^{\text{fdbk}(p,j)} - t_{j'}^p \\ &= (t_s^p + \tau_{s,j}^p + \delta(p,j) + \tau_{j,j'}^{\text{fdbk}(p,j)}) - (t_s^p + \tau_{s,j'}^p) \\ &= \tau_{s,j}^p + \delta(p,j) + \tau_{j,j'}^{\text{fdbk}(p,j)} - \tau_{s,j'}^p \,. \end{aligned} \tag{30}$$

As we observed before, $\theta_{j,j'}^p$ can be measured by j', provided that j' knows which congestion control feedback received from j is the first feedback sent following the arrival of p at j. Earlier, we showed that for a tree-based topology, $\theta_{j,j'}$ (which is the same as $E\{\theta_{j,j'}^p\}$) equals the difference between the round trip times of j and j'. In order to check the validity of a similar relationship here, let us first define the auxiliary parameter $\tilde{\tau}_{j'}^{p(j)}$, as

$$\tilde{\tau}_{j'}^{p(j)} \triangleq \tau_j^{p,j'} - \theta_{j,j'}^p \,. \tag{31}$$

From (29)–(31), it follows that

$$\tilde{\tau}_{j'}^{p(j)} = \tau_{s,j'}^{p} + \delta(\text{fdbk}(p,j), j') + \tau_{j',s}^{\text{fdbk}(\text{fdbk}(p,j),j')} \quad (32)$$

For comparison, $\tau_{j'}^{p}$ may also be expanded as

$$\tau_{j'}^{p} = \tau_{s,j'}^{p} + \delta(p, j') + \tau_{j',s}^{\text{fdbk}(p,j')} \ . \quad (33)$$

Note that the right hand sides of (32) and (33) have the common form $\tau_{s,j'}^{p} + \delta(m, j') + \tau_{j',s}^{\text{fdbk}(m,j')}$, where $m = p$ in (33), and $m = \text{fdbk}(p, j)$ in (32). The term $\delta(m, j')$ in this expression is the time taken by j' to send the first feedback after the arrival of message m at j'.

While the actual value assumed by $\delta(m, j')$ varies from message to message, we can make the following observation about its statistical average. If there is no statistical correlation between the arrival times of (data or feedback) messages at a receiver and the transmission times of feedback messages by the receiver, then the statistical average of the quantity $\delta(m, j')$ is independent of whether m is a feedback message or a data packet. In other words, the expected value of $\delta(m, j')$ is the same for all messages m, regardless of whether m is a data packet or a feedback message.

Now consider the term $\tau_{j',s}^{\text{fdbk}(m,j')}$. The assumption we made for $\delta(m, j')$ also implies that the statistical average of $\tau_{j',s}^{\text{fdbk}(m,j')}$ is independent of whether m is a feedback message or a data packet. Based on these observations, we conclude that while $\tilde{\tau}_{j'}^{p(j)}$ and $\tau_{j'}^{p}$ are not equal, they have the same statistical average. We thus have,

$$E\{\tilde{\tau}_{j'}^{p(j)}\} = E\{\tau_{j'}^{p}\} = \tau_{j'} \ . \quad (34)$$

In the special case of a tree-based topology where each receiver j has only one parent, $\tau_{j}^{p,j'} = \tau_{j}^{p}$. Therefore, $\tilde{\tau}_{j'}^{p(j)} = \tau_{j}^{p} - \theta_{j,j'}^{p}$, and the result in (34) agrees with our earlier observation in (18).

We now use the result in (34) to calculate the average round trip times τ_j. First, combining (28) and (31), we have

$$\tau_{j}^{p} = \min_{j' \in \mathcal{P}_j} \left(\theta_{j,j'}^{p} + \tilde{\tau}_{j'}^{p(j)} \right) \ . \quad (35)$$

All we need for a receiver j is the average round trip time $\tau_j = E\{\tau_j^p\}$, and not the round trip time τ_j^p for each packet p. Note that in general, the average of the minimum of several random variables is not equal to the minimum of their averages, i.e.

$$\begin{aligned} \tau_j &= E\{\tau_j^p\} \\ &\neq \min_{j' \in \mathcal{P}_j} E\left\{ \theta_{j,j'}^{p} + \tilde{\tau}_{j'}^{p(j)} \right\} \\ &= \min_{j' \in \mathcal{P}_j} \left\{ \theta_{j,j'} + \tau_{j'} \right\} \ , \end{aligned} \quad (36)$$

where the second equality follows from (34). We conclude that in order to determine the average round trip time τ_j, there is no way but to first find the packet round trip times τ_j^p (or similar per-packet quantities to be discussed below) for several packets, and then calculate their average. The only exception is when receiver j has only one parent j', in which case we can conclude from (35) and (34) that

$$\tau_j = E\{\tau_j^p\} = \theta_{j,j'} + \tau_{j'} \ , \quad (37)$$

which is identical to (18).

A recursive algorithm for the estimation of packet round trip times τ_j^p would have been provided by Equation (35), if we could replace the quantities $\tilde{\tau}_{j'}^{p(j)}$ in the right hand side of (35) with $\tau_{j'}^{p}$. Unfortunately, since $\tilde{\tau}_{j'}^{p(j)} \neq \tau_{j'}^{p}$, the packet round trip times to various receivers in the multicast group cannot be determined recursively. In the following, we exploit the fact that $E\{\tilde{\tau}_{j'}^{p(j)}\} = E\{\tau_{j'}^{p}\}$, and come up with an auxiliary quantity, called the modified packet round trip time $\hat{\tau}_j^p$. This quantity can be determined recursively and then used to calculate the average round trip time τ_j.

Definition 1 *The modified round trip time of packet p for receiver j is recursively defined as*

$$\hat{\tau}_{j}^{p} \triangleq \min_{j' \in \mathcal{P}_j} \left(\theta_{j,j'}^{p} + \hat{\tau}_{j'}^{p} \right) \ , \quad (38)$$

where

$$\hat{\tau}_{s}^{p} \triangleq 0 \ . \quad (39)$$

Theorem 1 *Assume that the transmission delays between the various pairs of receivers in the multicast group are statistically independent. Also assume that the waiting times $\delta(m, j)$ between the arrival of a data packet or feedback message m at a receiver j and the transmission of the first feedback from j after this arrival are statistically independent of the arrival time of m and the type of m. Then, for each receiver j, the modified packet round trip time $\hat{\tau}_j^p$ and the actual packet round trip time τ_j^p are identically distributed. In particular,*

$$E\{\hat{\tau}_j^p\} = E\{\tau_j^p\} = \tau_j \ . \quad (40)$$

Proof: Omitted.

The above definition and theorem demonstrate how a recursive algorithm may be used to efficiently estimate the average receiver round trip times τ_j in a multicast group with multi-parent hierarchical feedback topology. The main features of such an algorithm are outlined below:

- The algorithm should try to determine the modified receiver round trip times $\hat{\tau}_s^p$, for a sufficiently large number of packets p. The average receiver round trip times τ_j, can then be estimated by applying an appropriate averaging algorithm on the values of $\hat{\tau}_s^p$, obtained for the various packets p.

- The modified round trip times $\hat{\tau}_s^p$ need not be determined for every packet p. Instead, the algorithm could determine $\hat{\tau}_s^p$ for a subset of the packets transmitted by the sender that are specially marked by the sender for this purpose. Using this selective approach, the number of packets for which the modified round trip times are being measured concurrently, can be limited to a small number. This will minimize the state that needs to be stored at each receiver for calculating RTTs.

- Once a marked packet is received, its sequence number should be included in the next congestion control feedback sent by the recipient.

- When a receiver j has received a marked packet p and a congestion control feedback from a child k containing the sequence number of p, it determines the value of $\theta_{k,j}^p$ as the time elapsed between the arrival of p at j and arrival of the corresponding feedback from k at j.

- For each marked packet p, once a receiver j has determined its own modified RTT $\hat{\tau}_j^p$ (to be described below), it calculates the quantity

$$\hat{\tau}_k^{p,j} \triangleq \hat{\tau}_j^p + \theta_{k,j}^p \tag{41}$$

  for each child k and sends the result down to k. Obviously, this may be done only after $\theta_{k,j}^p$ has been determined. The sender initiates this procedure by sending down to each child k the value $\hat{\tau}_k^{p,s} = \theta_{k,s}^p$.

- For each marked packet p, each receiver j waits until it receives the quantities $\hat{\tau}_j^{p,j'}$ from all parents j'. It then computes its modified RTT for packet p as

$$\hat{\tau}_j^p = \min_{j' \in \mathcal{P}_j} \hat{\tau}_j^{p,j'}. \tag{42}$$

  As explained earlier, j can now calculate its own RTT τ_j by taking a running average of the quantities $\hat{\tau}_j^p$.

We note that this algorithm differs from the algorithm described in Section 4.1 for the tree-based topology in two ways. The first difference is that each parent must provide delay information for every marked packet to its children. This is in contrast to the previous algorithm where the delay information provided by each parent was based on running averages. The second difference is that each receiver gets delay information from multiple parents, and is responsible for computing the minimum over this delay information.

Extending Multi-Parent to Hop-Scoped: We now show a simple way in which the scheme for the multi-parent feedback topology can be extended to the hop-scoped feedback topology described earlier. The basic idea is to super-impose a multi-parent hierarchical feedback topology on the hop-scoped feedback topology. Let h be the maximum number of hops a feedback message travels. For each receiver, an appropriate value for h (such that the receiver remains connected to the sender) can be calculated using periodic session messages as described in [LESZ98].

The RTT measurement scheme works the same way as in the multi-parent hierarchy except the following. Instead of sending congestion control feedback to all parents, each receiver now sends congestion control feedback to all its neighbors within h hops. If the sender lies within h hops, the receiver sends its feedback to the sender only. Note that this process as-is could lead to feedback loops, which will not allow us to superimpose a multi-parent feedback topology. This is because the multi-parent feedback topology is loop-free. Therefore, we use the following technique to eliminate feedback loops.

For any receiver j, let $F(j)$ be the set of receivers from which j receives congestion control feedback. Then, if j is not the sender, it ignores congestion control feedback from all receivers $k \in F(j)$ where the IP address of k is lexicographically less than that of j. In

other words, receiver j (if it is not the sender) treats all neighbors with lexicographically lower IP addresses as its parents and all other neighbors as its children. Hence, for calculating updates to $\hat{\tau}_j^p$, $\theta_{k,j}^p$, and τ_j, it uses updates from receivers in $F(j)$ with lexicographically lower IP addresses, and sends it own updates to receivers in $F(j)$ with lexicographically higher addresses. However, if j is the sender, then it treats all its neighbors within h hops as its children in the superimposed multi-parent hierarchy and accepts congestion control feedback from all of them. There is still one problem with this scheme: if receiver j has no neighbors within a distance of h hops with an IP address that is lexicographically less than its own, its congestion control feedback will never reach the sender. To remedy this, we modify the algorithm as follows: during the exchange of session messages, the value of h is set at receiver j as

$$h = \min(h_m, h_s) \tag{43}$$

where h_s is the hop count to the sender and h_m is the minimum number of hops such that there is at least one receiver with an IP address lexicographically less than that of receiver j within h_m hops[2]. If $h_m \geq h_s$, j sends its congestion control feedback directly to the sender and no one else. The rest of the algorithm is the same as in the multi-parent case.

5 Loss Detection and Timeouts

In any reliable end-to-end transport mechanism, detection of lost packets is an important issue. Packet losses are the only indication of network congestion since the current Internet does not provide explicit congestion indication[3]. Loss detection for congestion control can be explicit or implicit. In the first case, individual lost packets are detected using sequence numbers. Every packet is labeled with a unique sequence number that is incremented whenever a new packet is sent. When a packet is received out of sequence, it indicates a potential loss. In the case of implicit loss detection, individual packet losses may not be detected: typically, some other parameter that is affected by packet loss (e.g., the packet receive rate) is calculated at the receiver and compared with the value of the same parameter when there is no loss (e.g., the rate at which the sender is actually transmitting, encapsulated in the packet header). If the difference between the two values exceeds a certain threshold, a packet loss is signaled. In this case, the receiver does not actually detect which individual packets have been lost, only that some packets have been lost, and (possibly) the number of lost packets. For multicast congestion control, either mechanism can be used.

An important component of the loss detection mechanism is the timeout mechanism at the sender. A timeout occurs at the sender when it has not received feedbacks for a sufficiently long period of time. This is an indication of lost packets, and therefore, potential network congestion. The timeout mechanism is responsible for deciding when a timeout occurs, which is directly related to the feedback topology since the feedback topology determines how the feedbacks get back to the sender.

We describe a timeout mechanism for the tree-based feedback topology first. In this case, as described in Section 4.1, each receiver j in the feedback tree sends a feedback every Δ_j units. Thus, if $d_1, d_2, \cdots, d_k$ denote the direct descendants of the sender in the feedback tree, the sender receives direct (aggregated) feedbacks from each of them every $\Delta_{d_1}, \Delta_{d_2}, \cdots, \Delta_{d_k}$ units, respectively. The sender sets the timeout for receiver d_i to some multiple (greater than 1) of Δ_{d_i}. Using a timeout that is greater than Δ_{d_i} ensures that there are not too many premature timeouts. However, this is not enough because of the *unreported loss* problem: each receiver j in the feedback tree keeps track of the last feedback received from each of its children. These feedback values are used to compute the consolidated feedback that is sent up the tree. Now, if feedbacks from j's child k start getting lost due to congestion, the sender will not be able to detect this because j will continue to use the value of the last feedback received from k. To avoid this problem, each receiver j maintains a local *epoch number* e_j that is incremented by Δ_j every time it sends a new feedback. For receiver j, let C_j be the set of all j's children. Then the epoch number E_j included by j in each of its feedback messages is given by:

$$E_j = min(e_j, min_k\{E_k : k \in C_j\}) \tag{44}$$

[2] Strictly speaking, the number of hops h should be given by $h = \max(h_c, \min(h_m, h_s))$, where h_c is the value for hop count calculated by the algorithm described in [LESZ98]

[3] This assumption is strictly true only for networks where the sole cause of packet loss is buffer overflow in routers. In wireless networks, where packet losses can occur due to data corruption, such an assumption is invalid.

where E_k is the epoch number j receives from its child k. If the epoch number received by the sender from the same direct descendant d_i does not change for a sufficiently long period of time, [4] it implies that there is at least one node in the subtree rooted at d_i that has not sent in a feedback for more than one feedback cycle. Therefore, this is also a signal for potential network congestion.

We now explain the rationale behind how the epoch numbers are incremented: receivers that send feedback less frequently (low frequency receivers) than others (high frequency receivers) will also increment their epoch number less frequently. Therefore, if the epoch numbers are incremented by the same amount at all receivers, the epoch numbers at the low frequency receivers will always be lower. This, combined with equation (44) implies that the epoch number for a low frequency receiver has a higher probability of being equal to the final consolidated epoch number that the sender sees. Now, let j, k be receivers with last reported epoch numbers E_j, E_k, respectively. Let j be a low frequency receiver and k be a high frequency receiver with $E_j \ll E_k$. Then the consolidated epoch number at the sender is at most E_j. Assume that the feedbacks sent by k after the one with epoch number E_k are lost. The sender will not detect this till the consolidated epoch number at the sender reaches E_k, which is undesirable. To circumvent this, the low frequency receivers must increment their epoch numbers by a higher amount, and the increment value of Δ_j encapsulates precisely this.

We now come to the broadcast-based feedback topology. In this case, for every packet sent, the sender receives feedback from the subset of receivers that did not receive any feedback during their random wait interval (see Section 4.2). For those receivers that belong to this subset, the sender is able to estimate the time it takes for them to send a feedback. However, such estimates may not be enough to set a timeout period since the receivers that were forced to suppress their feedbacks obviously require longer to send their feedback. Thus, if the timeout period at the sender is set using only the received feedbacks, it is possible that there is a receiver with a longer round-trip time, which implies that the sender may timeout prematurely. To be on the conservative side, it is necessary to set the timeout period to be large enough such that the chances of a premature timeout are low. Since such a timeout can be set only if we have a large number of observations for feedback times (that may not always be available), a default large value (such as 500 ms) should be used.

Finally, we consider the hop-scoped feedback topology. We have shown in Section 4.3 that a multi-parent hierarchy can be superimposed on a hop-scoped feedback topology. We therefore only show how to solve the timeout problem for the multi-parent hierarchy. The algorithm in this case is exactly the same as in the tree-based case except that now the periodic feedbacks containing epoch numbers are sent to all parents in the feedback topology.

6 Conclusion and Future Work

In this paper, we have presented alternative architectures for the construction of reliable multicast congestion control schemes. We have identified the three main architectural elements as the mechanism for feedback consolidation, the mechanism for RTT estimation and the mechanism for loss detection and timeouts. We have shown how each of these mechanisms can be implemented under different scenarios determined by the choice of the regulation parameter and the feedback topology.

For estimating round-trip times, we have presented two algorithms that are scalable and do not require the sender and receiver clocks to be synchronized. Of the three feedback topologies that we have discussed, RTT estimation works best for a tree-based topology. While it is difficult (if not impossible) to estimate round-trip times in the broadcast-based case without some form of clock synchronization, we are able to do so in the hop-scoped case by superimposing a hierarchical structure (similar to a tree) on the hop-scoped feedback topology. The main lesson here is that if RTTs have to be estimated in a scalable manner without using clock synchronization, the receivers in a multicast group must be organized in some form of hierarchy. The accuracy of the estimated timeouts depends on how well-defined this hierarchy is. Creating and maintaining such a hierarchy, however, has some overheads. The advantage of

[4] This period of time can be set equal to the maximum of $\Delta_j + \delta\tau_{j,j'}$ for all receivers j in the subtree rooted at d_i, where $\delta\tau_{j,j'}$ is the maximum expected variation in the transmission delay from receiver j to its parent.

broadcast-based schemes, at first glance, is that they do not have any such overheads. However, such schemes require periodic exchange of session messages which can also be used for estimating RTTs (among other things). Such messages typically tend to be all-to-all broadcast messages and may not scale well. An actual quantification of the tradeoffs between the overheads of session messages and the creation and maintenance of receiver hierarchies is left as the subject of further study.

We have also studied techniques for feedback consolidation for both tree-based and broadcast-based topologies. In a tree-based feedback topology, scalable consolidation of feedback is readily possible for both rate-based and window-based schemes. In contrast, in a broadcast-based feedback topology a limited degree of scalability can be achieved by using some form of feedback suppression. We have described two techniques for feedback suppression: one technique exploits knowledge about how the transmission rate is updated on the receipt of feedbacks to suppress unnecessary feedbacks. The second scheme is based on introducing random feedback delays which depend on the value of the feedback. We have shown that both of these techniques are mainly applicable to rate-based congestion control and have limited, if any, effectiveness in the case of window-based congestion control. Thus, for a broadcast-based topology, a limited degree of scalability is possible only for rate-based congestion control, whereas a tree-based topology provides completely scalability of feedback consolidation for both rate-based and window-based congestion control.

Finally, we have described schemes for detecting losses as well as timeouts. Although the timeout mechanisms that we describe are sender-based, they scale well since they do not involve feedback from all receivers. A more careful analysis of the tradeoffs between feedback overhead and accuracy of timeouts is required in order to determine the optimal feedback frequency.

To sum up, we can say that tree-based feedback topologies show better scaling properties than broadcast-based topologies. However, for small multicast group sizes, broadcast-based topologies could also be used, since the overheads of session messages in such cases may not be excessive. It is our belief that the actual choice may ultimately depend on the characteristics of the application that will use the multicast service.

References

[BG98] A. Basu and S. J. Golestani. Estimation of receiver round trip times in multicast communications. Technical report, 1998. Presented at the meeting of Internet Reliable Multicast Research Group, Virginia, AR, Dec. 1998. http://www.bell-labs.com/user/golestani/rtt.ps.

[Chi98] D. M. Chiu. Congestion control using dynamic rate and window. Technical report, 1998. Presented at the meeting of Internet Reliable Multicast Research Group, Arlington, VA, Dec. 1998.

[FJM+95] S. Floyd, V. Jacobson, S. McCanne, C-G. Liu, and L. Zhang. A reliable multicast framework for light weight sessions and application level framing. In *Proc. ACM SIGCOMM'95 Conf.*, pages 342–356, 1995.

[GS99] S. J. Golestani and K. Sabnani. Fundamental observations on multicast congestion control in the internet. In *to be presented at Infocom'99*, 1999. also see http://www.bell-labs.com/user/jamal/index.htm l.

[HF98] M. Handley and S. Floyd. Strawman specification for TCP friendly multicast congestion control. Technical report, 1998. Presented at the meeting of Internet Reliable Multicast Research Group, Arlington, VA, Dec. 1998.

[LESZ98] C. Liu, D .Estrin, S. Shenker, and L .Zhang. Local error recovery in SRM: Comparison of two approaches. *Submitted to IEEE Transactions on Networking.* http://catarina.usc.edu/estrin/papers/infocom98/local.ps.

[PFTK98] J. Padhye, V .Firoiu, D .Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *Proc. ACM SIGCOMM'98 Conf.*, pages 303–314, 1998.

[PSLB97] S. Paul, K. K. Sabnani, J. C. Lin, and S. Bhattacharyya. Reliable multicast transport protocol. *IEEE Journal on Selected Areas in Communications*, 15(3):407–421, April 1997.

[Rhe98] I. Rhee et al Rate-based fair multicast congestion control. Technical report, 1998. Presented at the meeting of Internet Reliable Multicast Research Group, Arlington, VA, Dec. 1998.

[RVC98] L. Rizzo, L. Vicisano, and J. Crowcroft. TCP-like congestion control for layered multicast data transfer. In *IEEE INFOCOM'98, the Conference on Computer Communication, San Francisco, USA, March 29–April, 1998*, 1998.

[WC98] B. Whetten and J. Conian. A rate based congestion control scheme for reliable multicast. Technical report, 1998. Presented at the meeting of Internet Reliable Multicast Research Group, Arlington, VA, Dec. 1998. http://www.gcast.com/docs/RateBasedCongestionControl.PDF.