# Scoped Hybrid Automatic Repeat reQuest with Forward Error Correction (SHARQFEC)

Roger G. Kermode[1]
Motorola, Chicago Corporate Research Laboratories,
1301 East Algonquin Road, MS 2712,
Schaumburg, IL, 60196-1078, USA
+1 (847) 538 4587
kermode@ccrl.mot.com

## 1. Abstract

**Reliable multicast protocols scale only as well as their ability to localize traffic. This is true for repair requests, repairs, and the session traffic that enables receivers to suppress extraneous requests and repairs. We propose a new reliable multicast traffic localization technique called Scoped Hybrid Automatic Repeat reQuest with Forward Error Correction (SHARQFEC). SHARQFEC operates in an end-to-end fashion and localizes traffic using a hierarchy of administratively scoped regions. Session traffic is further reduced through the use of a novel method for indirectly determining the distances between session members. For large sessions, this mechanism reduces the amount of session traffic by several orders of magnitude over non-scoped protocols such as Scalable Reliable Multicast (SRM). Forward Error Correction is selectively added to regions which are experiencing greater loss, thereby reducing the volume of repair traffic and recovery times. Receivers request additional repairs as necessary. Simulations show that SHARQFEC out performs both SRM and non-scoped hybrid Automatic Repeat reQuest / Forward Error Correction protocols. Assuming the widespread deployment of administrative scoping, SHARQFEC could conceivably provide scalable reliable delivery to tens of millions of receivers without huge increases in network bandwidth.**

### 1.1 Keywords

Multicast, reliable, scalable, ARQ, FEC, hierarchy, administrative-scoping.

## 2. Introduction

Recent advances in microprocessor power have realized the concept of digital convergence at the source and receiver. The manipulation and presentation of traditionally analog media, such as audio, images, and video, can now be done digitally. Furthermore, this ability is becoming increasingly affordable - to the point where the equipment needed is finding its way into the home in the guise of cheap home computers, DVD players, and television set top boxes.

However, digital convergence in the channel is not so advanced. Thanks in no small part to the explosive popularity of the World Wide Web[1], the Internet Protocol (IP) [2, 13] is assured of being the primary means for delivering information over computer networks for the foreseeable future. The current suite of protocols used to deliver information over IP are primarily used in a point-to-point fashion. This mode does not mesh well with the point-to-multipoint, and multipoint-to-multipoint, delivery modes one normally associates with the distribution of audio and video traffic.

Multicast transport protocols deliver data so that only one copy of each packet traverses over any given link in the tree (or trees) that joins the members of a multicast group. Unicast protocols can emulate multicast ones by establishing multiple connections, but at the expense that each packet must be resent for every member of the group. Multicast transport protocols provide a useful service for "group" applications since they reduce network traffic considerably when there are large numbers of participants.

While IP does support multicast, it does so in the same best effort manner it uses for unicast traffic; that is, reliable delivery is not guaranteed. The lack of a reliability mechanism that scales within IP multicast is a key factor that limits its widespread commercial use. Computer programs and legal documents must be delivered without loss for them to have any utility. Likewise, one can also make the argument that use of highly compressed representations such as MPEG-2 [8] also mandate the inclusion of reliability mechanisms, since losses now become more perceptible and longer lasting. For these reasons, much research has been devoted to realizing a scalable reliable multicast protocol for use on the Internet. Some of the methods developed have been quite ingenious.

The first class of methods developed can be grouped under the banner of Automatic Repeat reQuest (ARQ). Scalable Reliable Multicast (SRM) by Floyd *et al.* [3] popularized the notion of allowing receivers to effect repairs as well as the sender, and introduced redundant traffic suppression through the use of random delays that are proportional to a receiver's distance to the sender. Others break the distribution tree up into parts, and use staged delivery schemes that delegate responsibility to specialized receivers that serve a portion of the tree below. The Tree-based Multicast Protocol (TMTP) by Yavatkar *et al.* [21], the Reliable Multicast Transport Multicast Protocol (RMTP) by Paul *et al.* [12], and Log-based Receiver Reliable Multicast (LBRRM) by Holbrook *et al.* [6] are examples of these types of protocols. Papadopolous *et al.* proposed that the ability to direct traffic to certain parts of the trees be incorporated into the routers so that they can selectively multicast out of one interface and not others [11].

Recently, methods based on merging Forward Error Correction (FEC) with ARQ have been gaining favor [4, 7, 10, 14, 15, 16]. Protocols based on these techniques work under the assumption that losses in different parts of the network are uncorrelated. This

[1] The work presented in this paper was performed by the author at the MIT Media Laboratory as part of his Ph.D. and was supported in part by the members of the Singapore Program in Digital Media and Digital Life Consortium, AT&T, and Motorola.

assumption is borne out through observation of losses in the Multicast Backbone (MBone) [5, 20]. Given this premise, these techniques transmit data packets in groups and then effect repairs by sending repair packets constructed from the original data. Typically, packet groups are constructed by taking $k$ data packets and constructing an additional $h$ repair packets, where $h$ is usually much smaller than $k$.

A number of techniques exist to determine how many repair packets to send, and when to send them. In [7] Huitema explored the application of FEC as separate protocol layer placed beneath an ARQ reliable multicast layer. Nonnenmacher *et al.* compared a number of schemes in [10]. They first examined a coarse grained "layered" approach in which $h$ was fixed and any unrecoverable data packets were placed in the next group. This scheme was then compared with two fine grained "integrated" schemes. In the first integrated scheme, $h$ was set to a large enough value to ensure delivery to all receivers and receivers left the multicast group after receiving sufficient packets to reconstruct the group. In the second scheme ARQ was added, with receivers requesting additional repair packets as needed. Their work showed that the second integrated scheme achieved the lowest network overhead.

Each of the protocols just mentioned provides a reliable mechanism for the delivery of traffic to more than one receiver. However, varying degrees of difficulty would be experienced by all if they were used to deliver a large newspaper to a million subscribers or for a live sporting event such as the Super Bowl, or a World Cup Soccer final. Protocols that rely on receivers to have an estimate of the distances between itself and each other receiver would break, because for sessions of $n$ receivers, each receiver must listen to traffic volumes that increase as $O(n^2)$ and maintain state that increases as $O(n)$. Others that rely on specific designated receivers or router modifications require wide spread deployment for their benefits to be realized. In addition, solutions that rely only on selected designated receivers tend to limit the opportunities for repairers since not all nodes can participate in the repair of their peers. Hybrid ARQ/FEC methods fare better, but still subject every receiver to the repair traffic needed to correct the losses of the worst.

In each of these cases the feature needed to make them work is the same:

- A means for local recovery that is deployable in a scalable fashion over the existing network with minimal, preferably no, modifications to the routers themselves.

We now show how the judicious use of administrative scoping to create hierarchy of nested regions combined with a modified hybrid ARQ/FEC algorithm provides this missing functionality.

# 3. Administratively Scoped Localized Recovery

## 3.1 Existing non scoped techniques

To understand SHARQFEC's design, it is easiest to start with a simple example and examine the traffic flow generated by standard ARQ and nonscoped FEC protocols. Consider the delivery tree shown in the top of Figure 1. A single source at the root of the tree delivers data to a number of receivers. Different branches in the tree are subject to loss at different rates, with some being virtually lossless and others experiencing considerable losses due to congestion. The total loss between the source and a given node is calculated by compounding the loss rates of each link between the source and that node

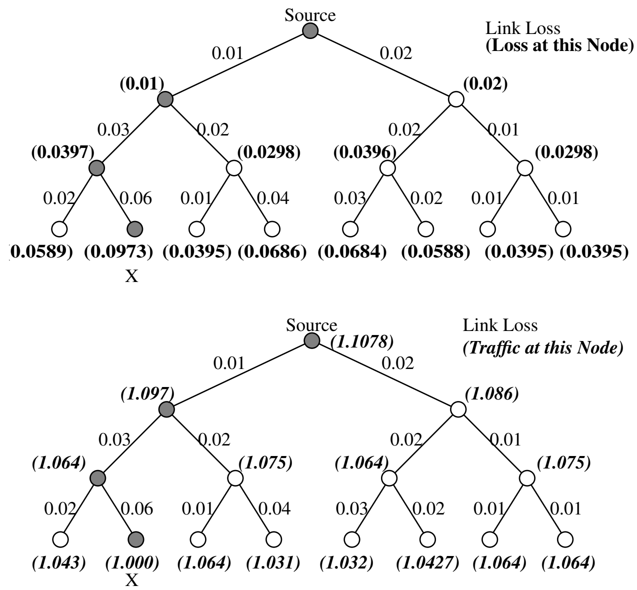


**Figure 1. Example Delivery Tree / Normalized traffic Volume for Non-Scoped FEC.**

$$\text{Total Loss} = 1 - \prod \Pr(\text{No loss}) \forall \text{links from source to node}$$

Furthermore, if one assumes that losses are independent, the probability that all receivers will receive a given packet is

$$\Pr(\text{all nodes receive a given packet}) = \prod \Pr(\text{No Loss}) \forall \text{links in the delivery tree}$$

For the tree shown in Figure 1, this equation yields a probability of 27.0%. In other words, every time the source multicasts a packet there is a better than 70% probability that at least one receiver will fail to receive it. In real world situations, where there are many more receivers connected by trees containing considerably more links, this figure has been shown to be close to 100% [5]. This simple analysis highlights the global nature of simple non-scoped ARQ techniques. Either the source must transmit each packet multiple times, or, if local recovery is being used, every session member must participate in the repairs of every other receiver. This problem is exacerbated when the repairs themselves are lost.

The other popular approach for effecting repairs has been to send packets in groups and to then send additional packets created with Forward Error Correction (FEC) techniques. As each FEC packet has been generated from the original packets in the group, it can repair the loss of any one packet. Multiple missing packets are repaired using multiple FEC packets. This method reduces the volume of repair traffic considerably. Now the sender can add just enough redundancy to compensate for the greatest loss experienced by any one receiver.

Unfortunately, losses tend to be unevenly distributed, making it difficult to determine how much redundancy is needed. Consequently, receivers experiencing higher losses usually end up requesting sufficient repairs to cover their own losses. Thus, the additional FEC traffic needed to repair this one receiver is sent needlessly to receivers experiencing lower loss. The tree on the bottom of Figure 1 shows the normalized traffic volume seen at each node when the source adds enough redundancy to compensate for receiver X, which experiences 9.73% loss.

### 3.2 Local Recovery through administrative scoping

The ability to restrict the scope of repair traffic is a key tool for increasing the scalability of reliable multicast protocols. In the previous section, it was shown that even hybrid ARQ/FEC techniques result in additional traffic on lightly congested links and on links near the source. This task can be achieved through the use of a hierarchy of nested administratively scoped regions.

This mechanism has two advantages over other hierarchical mechanisms, such as the installation of dedicated receivers [6, 12] or modifying the routers to support selective multicast [11]. First, the system must be robust without being inefficient. A reliance upon specially configured receivers creates the potential for catastrophic failure.

In addition, hierarchical schemes that elect designated receivers dynamically, and then use combinations of Time-to-Live (TTL) scoping and multiple multicast groups, have the potential to be inefficient since traffic still has the potential to seep outside the hierarchy. Second, the resulting system must be deployable in a quick, easy, and scalable fashion. Protocols that require router modifications, while possibly being the most efficient, become much less efficient when run on unmodified routers. Furthermore, these protocols have a tendency to create additional state in the router, and therefore are subject to some concern about their ability to scale when numerous sessions are present. This concern is magnified for routers in the backbone.

The creation of a hierarchy of administratively scoped zones affords greater localization through several means. Consider the example in Figure 1 from the previous section, this time overlaid with a three level hierarchy of zones as shown in Figure 2. In this scheme, there is a single data channel with maximum scope, and an additional repair channel for each zone. Data is transmitted over the data channel and repairs are selectively added within each zone to compensate for the losses between the current zone and the next zone with the highest loss rate. Thus, the source in Figure 2 need only add sufficient redundancy to guarantee delivery of each group to receiver Y, which will in turn add just enough redundancy to ensure delivery of each group to receiver Z.

As the loss rates will vary over time, the level of redundancy will change from group to group. Should too much redundancy be injected at one level in the hierarchy, receivers in subservient zones will add less redundancy. Should greater losses occur than expected, receivers will use ARQ to request additional FEC packets as needed from larger scoped zones. Receivers perform suppression on all NACKs as appropriate.

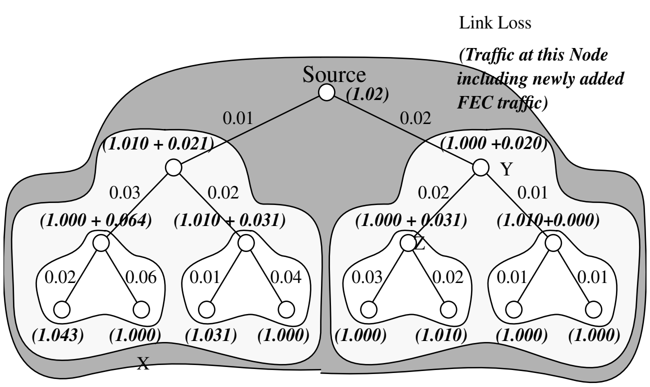


**Figure 2. Redundancy Injection using FEC**

This scheme requires the election of a representative for each zone. This receiver, called a Zone Closest Receiver (ZCR), should be the zone's closest receiver to the ZCR of the next-largest zone. While the existence of ZCRs does create singular points of failure in the distribution tree, the ability of receivers to increase the scope of their NACKs without reconfiguring the hierarchy minimizes the consequences of ZCR failure. When the scope is increased in this manner, all receivers within the higher scoped region participate, and not just the ZCR at the next highest level. Thus, the system merges the benefit of efficiency afforded by hierarchy with the robustness afforded by peer-to-peer recovery. As the process by which receivers determine the ZCR for each zone has implications for session traffic, the process for electing ZCRs is deferred for discussion later in Section 5.2.

## 4. SHARQFEC Algorithm Description

Packet groups are delivered in a two-phase process; a loss-detection phase and a repair phase. In the loss-detection phase, the source multicasts the original packets over the data channel and receivers announce the number of packets lost to the other receivers within their zone. Once all the packets within a group have been sent, the repair phase begins.

During the repair phase, receivers attempt to effect repairs as needed to other receivers within their zone. Repairs are effected in two ways. First, receivers may request them in a manner similar to ARQ methods, with the small modification that the NACK now indicates how many additional FEC packets are needed to complete the group and not the identity of an individual packet. Second, Zone Closest Receivers in each zone may automatically inject additional FEC packets into the stream without waiting for the NACKs to arrive. The number of FEC packets injected is determined from previous losses and decays over time. Should the number of additional FEC packets injected by a Zone Closest Receiver prove insufficient, a receiver may request further repairs using the first method.

The two-phase delivery process used by SHARQFEC senders and receivers is as follows:

Loss Detection Phase (LDP)

- The sender divides its packet stream into groups that are transmitted sequentially along with a small number of redundant FEC packets. (The decision as to how many redundant packets to send will be explained shortly.) After sending the last unrequested FEC packet, the sender automatically enters the repair phase.
- Receivers take note of the order in which packets arrive and maintain a count of how many original packets were lost in transit, the Local Loss Count (LLC). In addition, each receiver also maintains state about the maximum number of losses experienced by any one receiver in each of the $n$ administratively scoped repair zones of which it is a member. The maximum LLC for a zone $n$ is known as the Zone Loss Count, ZLC($n$).
- Upon the reception of a packet in a new group, receivers estimate the time by which all packets within the group should be received. They then set a Loss Detection Phase (LDP) timer to expire at this time. This estimate is calculated on the basis of the inter-packet arrival time, which is refined on a group-by-group basis. Initially, the inter-packet arrival time is itself estimated using the advertised channel bandwidth and the advertised size of the data packets.

- When a receiver detects the loss of one or more original packets, it increments its LLC according to the number of missing packets detected. Should the LLC exceed the ZLC for the current scoped repair zone, it starts a request timer in the same manner as specified by the SRM protocol with fixed timers. That is, delay is chosen on the uniform interval $2^i[C_1 d_{S,A}, (C_1 + C_2)d_{S,A}]$, where $C_1 = 2$, $C_2 = 2$, and $d_{S,A}$ is the receiver A's estimate of the one-way transit time for traffic between the source, S, and itself. The $i$ variable is initially set to 1 and is incremented by 1 every time a NACK that does not increase the appropriate ZLC(n) is received. Any time a repair arrives, $i$ is reset to 1.
- If a receiver's request timer expires, the receiver transmits a NACK containing its LLC (which will become the new ZLC for that scope zone), the greatest packet identifier seen, and the number of repair packets needed. NACKs are transmitted to the repair channel with scope equal to that of the smallest scope zone for a particular partition, subject to the restriction that if the originating source is a member of the smallest partition, the repair channel corresponding to the largest scope is used instead.
- Receivers will suppress NACKs if their LLC is less than or equal to the receiver's own estimate of the ZLC for a zone. NACK suppression will also take place should a receiver receive a NACK that increments the ZLC so that the LLC is less than or equal to the new ZLC value for a zone.
- When a receiver receives a NACK, it updates the appropriate ZLC and then checks to see if the NACK's last received packet identifier causes the detection of any further lost packets. If additional packets (either original or FEC) are missing, the receiver follows the operations outlined immediately above.
- In addition to using received NACKs to update the ZLC information, receivers also use NACKs to maintain information about the number of repairs needed by other receivers. This information is then used to speculatively schedule replies, with the expectation that by the time the timer expires a sufficient number of repairs will have been received to complete the packet group. Increases to the number of speculatively queued repairs do not reset the reply timer. The reply timer is set in the same manner as specified by the SRM protocol with fixed timers. That is, the delay is chosen on the uniform interval $[D_1 d_{S,A}, (D_1 + D_2)d_{S,A}]$, where $D_1 = 1$, $D_2 = 1$, and $d_{A,B}$ is the receiver B's estimate of the one-way transit time for traffic between the sender of the NACK, A, and itself. Note that the back-off mechanism used by SRM repair timers is omitted for SHARQFEC.
- Speculatively queued repairs are dequeued upon the reception of repair packets. As repairs from larger zones will also be received by any smaller administratively scoped zones known to a session member, speculatively queued repairs for smaller zones will be decremented as well.
- When a receiver sends a NACK, it includes the largest known packet identifier. Similarly, when a session member starts to send one or more repairs the member includes what will be the new highest packet identifier. These two measures are designed to minimize the likelihood of the same repair packet being sent twice by different repliers.
- When a receiver's LDP timer expires, or it receives enough packets to reconstruct the group, it enters the repair phase. A sender enters the repair phase immediately upon the completion of sending the original packets.

Repair Phase (RP)

- After transmitting the last data packet of the group, the sender enters the repair phase, immediately generating and transmitting the first of any queued repairs in the largest scope zone. Since this repair will also be received by any smaller administratively scoped zones known to the sender, the pending repair queues for these zones are shortened appropriately. The sender then starts its internal repair reply timer with a short interval designed to spread out any subsequent repairs. In the simulations reported later in 6.2, the interval was set to half that of the inter-packet interval observed for successive data packets.
- Upon the reception of packets sufficient to reconstruct the group, ZCRs likewise become repairers and generate and transmit the first of any additional queued repairs to the zone for which they are responsible.
- Once a non-ZCR receiver successfully receives sufficient packets to reconstruct all the packets within a packet group, that receiver becomes a repairer.
- Upon receiving a NACK, the sender and repairers update their state to reflect the number of outstanding repairs needed to complete the administrative scope zone defined within the request. They also update their maximum packet identifier to minimize the likelihood that any repair packets they subsequently transmit will not already have been sent by another receiver.
- Before sending any repairs, non-ZCR repairers perform suppression using a reply timer in the same manner mentioned earlier. As replies may consist of multiple repairs, a repairer will only cancel its reply timer when a sufficient number of repairs has been received to effect the entire repair.
- Should a repairer's reply timer expire before the reception of enough repair packets, it follows the same set of steps outlined above for senders entering the repair phase.
- Should a repairee detect that it has lost a repair and that further repairs will be needed, it transmits a new NACK indicating that it needs more packets, subject to the same suppression rules described in the Loss Detection Phase. The scope of successive attempts will be increased after two attempts at each zone to the next-largest scope zone, until the largest scope zone is reached.

One aspect of SHARQFEC's operation not covered by this set of rules is the determination of how many automatic repairs a ZCR should add to its zone upon having successfully received a group's packets. If one makes the assumption that losses (however bursty in nature) tend to vary slowly from group to group, then a receiver can estimate the number of automatic repairs to send by applying a simple exponential weighted moving average (EWMA) filter to the ZLC for previously received groups. As the automatic repairs will suppress NACKs in situations when the ZLC for the current group is less than the predicted ZLC, the EWMA filter will use the receiver's LLC in cases where no NACKs are received to indicate the true ZLC.

The coefficients used within the test simulations were as follows:

$$zlc_{predicted}(n) = 0.75 \times zlc_{predicted}(n-1) + 0.25 \times zlc(n)$$

if true ZLC known

$$= 0.75 \times zlc_{predicted}(n-1) + 0.25 \times zlc(n)$$

if true ZLC unknown

Another problem in estimating the ZLC for subsequent groups is knowing when to measure the ZLC for a group during the repair phase. Measure it too early, and the value might be too low, as a NACK might still be outstanding from a distant receiver. Measure it too late, and the measurement's accuracy will be lessened. This problem can be solved by noting that the maximum delay a receiver's NACK will experience is equal to the RTT time between itself and the ZCR plus the maximum delay due to its suppression timer. Thus, a ZCR is guaranteed of learning the true ZLC for a zone if it waits for a period equal to two and half times the Round Trip Time (RTT) between itself and the most distant known receiver.

## 5. Administratively Scoped Session Management

One of the major shortcomings of using SRM-like timers for suppression is the requirement that each session member have an estimate of the Round Trip Time (RTT) to every other member. This requirement results in $O(n)$ state per receiver where $n$ is the number of receivers in the session. However, the real problem is that $O(n^2)$ session traffic is required to maintain this state. Thus, attempts to scale beyond more than a few hundred members will result in catastrophic congestion.

The key to solving this dilemma is to realize that when a loss occurs, the ensuing NACK is likely to originate from a receiver near to the link on which the loss occurred. Furthermore, the receiver generating the repair is likely to be upstream from the receiver that sent the NACK. This means that it is more important for session members to know the RTTs to nearby members than distant ones. With this realization, it becomes possible to reduce the amount of session traffic and state by constructing a session mechanism that allows receivers to collect detailed information about other nearby receivers and a summarized view of more distant receivers.

Administratively scoped repair zones can assist in the reduction of this state information. Consider the figure below. Here, a cloud of receivers receives data from a single source, node 1, that subscribes only to the largest administratively scoped repair zone, Z0. Receivers below the source subscribe to one of two intermediate scoped zones: Z1 and Z2. Finally, the receivers farthest from the source subscribe to one of four locally scoped repair zones: Z3, Z4, Z5, and Z6.

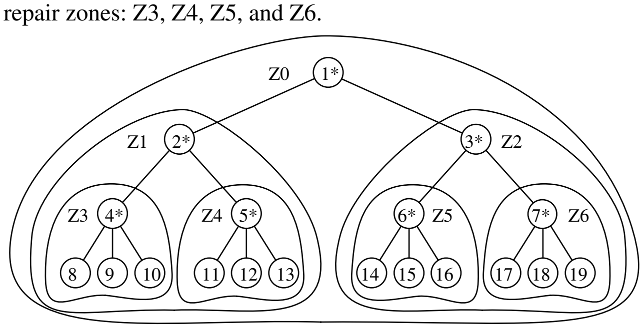


**Figure 3. SHARQFEC Hierarchical Scoping**

Now consider the receivers in zone Z4. Thanks to the SRM-like timers, Zone Z4's receivers are unlikely to receive repairs originating from receivers in zone Z3, and are even more unlikely to receive repairs originating from receivers in zone Z2. The majority of repairs generated in response to a Z4 receiver's NACK will originate from the source, other receivers within zone Z4, and receivers between the source and Z4. Thus, it is important that the receivers have very good estimates of the RTT to the source and other receivers within Z4, and good estimates to the receivers between the source and Z4. Estimates of the RTT to the remaining receivers are less important.

The observation that the accuracy in RTT estimates can vary according to the relative position of receivers affords several opportunities to reduce session traffic volume. First, one can aggregate estimates of the RTT for several receivers into a single estimate. For example, the receivers in zone Z4 need not know the RTT to every receiver in zone Z2. It is likely that an estimate of the RTT from node 5 to node 3 would be sufficient to perform suppression for a request/repair transaction made in the global scope zone Z0. Second, RTT measurement need not be done in a single step; instead, it can be broken up into several independent RTT measurements. For example, if one accepts that node 3 is the closest receiver in zone Z2 to the source, then the remaining receivers in zone Z4 could make an estimate of their own RTT to the source by adding the node 2's RTT to the source to their own estimate of the RTT to node 2.

SHARQFEC's session management exploits the opportunities this observation affords, by aggregating session traffic and limiting its scope according to the following set of rules:

- Each Zone determines the node closest to the data source to act as a representative for the receivers within that zone. This receiver is known as the Zone Closest Receiver (ZCR) and is determined either by design—in which case a cache is placed next to the zone's Border Gateway Router—or by a process that will be described shortly.
- A ZCR at the next-largest scope zone is known as a parent ZCR.
- Nodes other than the ZCR within a particular zone limit the scope of their RTT determination session traffic so that it is carried within the smallest-known scope zone. This enables them to determine the RTT between themselves and the other nodes within the scope zone.
- The ZCR for a particular zone participates in RTT determination for that scope zone, and also the next-largest scope zone (known as the parent zone).
- Nodes maintain state information about the RTT between themselves and other nodes within the smallest-known scope zone. In addition, they also maintain state about the RTT between the ZCR at each successively larger scope zone and the other nodes within that ZCR's parent zone.
- When a node sends non-session traffic that requires nodes hearing it to determine the RTT between the sending node and themselves, the sending node includes estimates of the distance between itself and each of the parent ZCRs that will hear the message.
- Nodes calculate the distance between themselves and their ancestral ZCRs by adding the observed RTTs between successive generations.
- Nodes that have just received a packet from another node, and need to calculate the RTT between themselves and that node, do so as follows: First, they examine the RTT information included in a packet to determine the largest scope zone for which they can find a match between sibling ZCRs. Next, they calculate the RTT between themselves and the sending node by adding the RTT between the sibling ZCRs to the RTT between themselves and their parent ZCR, and the RTT between the sending node and the appropriate ancestral sibling as extracted from the packet.
- Nodes shall randomly stagger their session messages to reduce the likelihood of convergence. (In the simulations that follow, the delay was chosen on the uniform interval $[0.9, 1.1]$ seconds. To speed up convergence, the delay for the first three session messages was chosen on the uniform interval $[0.05, 0.25]$ seconds.)

## 5.1 Indirect Round-Trip Time Determination

The application of the rules above to restrict session traffic via administrative scoping has a marked effect on both the amount of traffic and state that must be handled by each receiver. For example, at the largest scope zone in the example given in Figure 3, Z0, the source need only participate in RTT estimation between itself, the other session members that only subscribe to Z0, and the ZCRs of the next smallest zones, Z1 and Z2 (Figure 4). Similarly, in the intermediate scope, Z1, only those receivers whose smallest scope zone is Z1 and the ZCRs of the next smallest zones, Z3 and Z4, participate in the RTT process for that level. Finally, in the smallest zone, Z4, only those receivers who subscribe to Z4 take part in the process.

This system of session announcements results in the receivers having a complete set of information about all the other receivers within their zone, the receivers in the regions directly between their zone, and the largest scope ZCR receiver behind each obscured region. Figure 5 shows explicitly how this scheme results in a dramatic reduction in the amount of state that must be maintained by each receiver. Whereas before each receiver would have had to maintain state information for 18 other receivers, in this example the worst case is now 7 receivers. The reduction in state is more pronounced as the number of levels and fanout in the hierarchy increase.

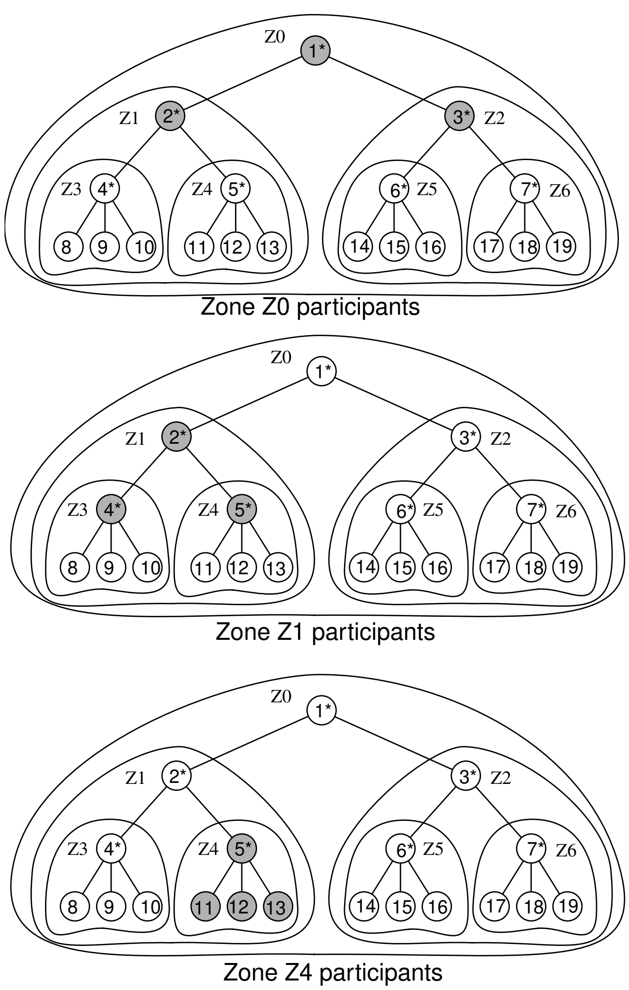


**Figure 4. Receivers involved in RTT determination phase in Zone Z0, Z1, and Z4 respectively**

To construct these reduced state tables, receivers selectively listen to and record the state traffic as follows. Starting with the smallest scope zone, each receiver constructs a state table with entries for each of the receivers that participate solely in that zone. Next, the receiver extracts the ZCR from that zone and listens to the session announcements for that receiver in the next-largest scope zone. Thus, in Figure 5, receiver 11—having determined that receiver 5 is the ZCR for the smallest scope—listens for session announcements originating from receiver 5 at the next-highest scope zone, and thereby determines the RTTs to receivers 2 and 4. This process is applied to all larger scope zones.

With these tables in place, receivers can estimate the RTT to any given receiver, provided that a message from that receiver contains its own estimate of the RTT to each of the ZCRs above it. Consider a packet originating from receiver 8 that is sent to the intermediate scope zone, Z1. Receivers 4, 9, and 10 will already have direct estimates of the RTT to 8, and receivers 2, 5, 11, 12, and 13 will not. However, if receiver 8 includes its estimate of the RTT between itself and receiver 4, these other receivers can compute an estimate by adding one or more RTTs together. For example, receiver 13 would add the entries for the RTTs between itself and receiver 5, receiver 5 and receiver 4, and the supplied RTT between receivers 8 and 4 to arrive at an estimate of the RTT between itself and receiver 8.

It should be noted that the indirect calculation in this manner is most accurate in situations where the ZCR is immediately adjacent to the border gateway router (BGR) that enforces administrative scoping between the ZCR's zone and its parent. If the ZCR is located away from the BGR, traffic from any receivers connected directly to the BGR will not pass by the ZCR. Consequently, the RTTs estimated by these receivers is greater than the actual RTT.

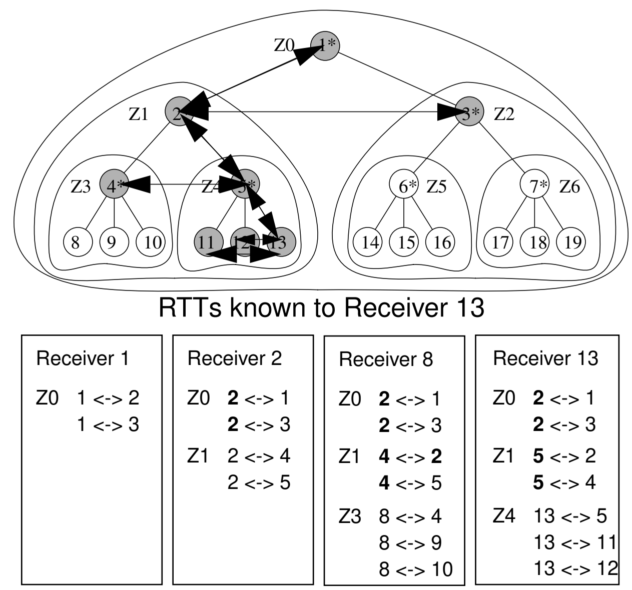


**Figure 5. Session State maintained by selected receivers**

$$d_{8,\,13} = d_{8,\,4} + d_{4,\,5} + d_{5,\,13}$$

Estimated From Received Packet From Internal Tables

**Figure 6. Indirect RTT estimation example**

Receivers that generate inaccurate RTT estimates do not generate traffic that breaks SHARQFEC's operation; in fact, the greater delays that measurements cause will result increased suppression.

Breaking up session management traffic into hierarchical overlapping regions results in a significant savings. The amount of traffic devoted to the determination of RTTs in each zone now drops from a constant $O(n^2)$ for all receivers to $O(\sum n_\alpha^2)$ where $n_\alpha$ is the number of session members participating at each of the $\alpha$ regions that are observable by a receiver. One notes two important characteristics in this new relationship. First, the farther a receiver is from the source, the more zones it will likely participate in, and therefore the more session traffic it will receive. And second, since the $n^2$ relationship is still present, efforts should be made to limit the number of receivers participating at each level.

The process of updating the internal tables of RTTs and electing the ZCRs at the various scope zones occurs periodically and is performed in a top-down fashion. ZCRs are elected at the largest scope zones first, with each smaller zone backing off until the parent zone has elected a ZCR. ZCR election occurs in two phases: the RTT determination phase and then an optional ZCR challenge phase. During the RTT determination phase members of a particular zone, exchange session messages that contain:

- the time at which the message was sent,
- the identifier of the ZCR for that zone,
- the recorded distance between the ZCR of that zone and the ZCR of the next-largest zone,

and a list with the following information about each receiver heard from that zone.

- The receiver's identity,
- Time elapsed since the last session message was received from the receiver,
- The sender's own estimate of the RTT between itself and the receiver.

Receivers that are the ZCR for a zone participate not only in that zone but that zone's parent as well, and as a result send out two session messages when their session timers expire. The first session message lists entries for the child zone's receivers and is sent to the child zone, while the second message is sent to the parent zone and lists the parent zone's receivers.

When a receiver receives a session message from a peer in its smallest-known scope zone, it records the time, the sender's timestamp, and the time the message was received for inclusion in its next session message. The receiver also scans the list of entries and if it finds an entry corresponding to its own identifier, it extracts the information and updates its estimate of the RTT to the sender. When a receiver receives a session message from its local ZCR, or one of its parents, it records that node's identifier and the RTT listed with each entry, thereby obtaining a record of how far the ZCR is from each of its peers. It should be noted that, thanks to the scoping mechanisms imposed, a receiver will only receive session messages from peers and the ZCRs of the largest obscured zones.

To see how SHARQFEC's scoped RTT estimation mechanism translates into savings in session state in the real world, consider the construction of a national distribution network to deliver a sporting event to 10,000,000 receivers using a 4 level hierarchy

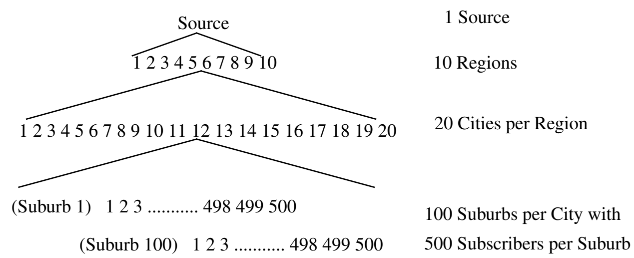


**Figure 7. Receiver distribution in a hypothetical National Distribution Hierarchy**

| | National | Regional | City | Suburb |
|---|---|---|---|---|
| Receivers / Zone | 0 | 1 | 1 | 500 |
| Number of Zones | 1 | 10 | 20 | 100 |
| Number of Receivers | 0 | 10 | 200 | 10,000,000 |
| RTTs maintained/ receiver | 10 | 30 | 130 | 630 |
| Ratio Scoped to Non-Scoped Traffic | 100 / $10{,}000{,}021^2$ | 500 / $10{,}000{,}210^2$ | 10,500 / $10{,}000{,}210^2$ | 35,5000 / $10{,}000{,}210^2$ |
| Ration of Scoped to Non-Scoped State | 1 / 1,000,021 | 3 / 1,000,021 | 13 / 1,000,021 | 63 / 1,000,021 |

**Figure 8. Receiver state reduction through the use of indirect RTT estimation**

consisting of 10 regions, with each region encompassing 20 cities having 100 suburbs with 500 subscribers in each. In order to assist with intermediate caching dedicated caching receivers have been distributed at each of the bifurcation points to act as ZCRs except at the suburb level where one of the 500 subscribers will be elected to perform this task.

In this example there is one sender and 10,000,210 receivers. When direct non scoped RTT estimation is used, the sender and receivers are required to transmit and maintain state for each of the 10,000,210 other session members. Clearly, direct non-scoped RTT estimation will not work. However, the table shows that the state (and hence traffic) that each receiver must keep track of is reduced to a much more manageable level

## 5.2 Adaptive Selection of Zone Closest Receivers

As receivers join and leave sessions, it is almost certain that the multicast distribution tree between the source and receivers will change. The ZCR challenge phase is designed to accommodate this change, and provides a means for the receivers within a zone to elect a new ZCR, should the old ZCR leave the session or should the distribution tree change so that it is no longer the closest receiver to the source for that zone. Service providers may configure their networks so that there is a static ZCR adjacent to the router that enforces the boundary between the local-scope zone and the parent-scope zone. In these cases, the ZCR challenge phase will only be necessary should one wish to provide robustness in the event that the dedicated receiver ceases to function.

Like the RTT determination phase, the ZCR challenge phase makes the assumption that the distribution trees created by the routers do not differ significantly near the boundaries between regions of different scopes, or as a function of the source of the packet. For the most, part this assumption holds, since the majority of source-based optimizations made by multicast routing protocols tend to occur at the leaves of the tree and not within the backbone of the network. To assist in understanding of the ZCR challenge phase Figure 9 shows the four possible challenge cases. Node 1 in both cases is the current ZCR, with nodes 2 and 4 closer to the parent ZCR, and nodes 3 and 5 farther away. For the purposes of the following discussion, it is assumed that the transit times for messages between two receivers is the same as the distance shown.

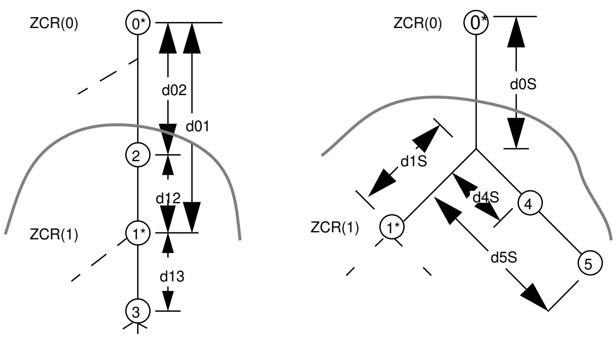


**Figure 9. The Four Possible SHARQFEC ZCR Challenge cases**

Consider first the chain example shown on the left. Here, node 1 is currently the ZCR for the zone containing nodes 1, 2, and 3. Now, let node 1 transmit a ZCR challenge to the parent ZCR, node 0. This challenge will be received by nodes 0, 2, and 3 at $d_{01}$, $d_{12}$, and $d_{13}$ respectively. After a processing delay, node 0 transmits a ZCR response containing the delay between when the ZCR challenge was received and the ZCR response was sent. Assuming this delay to be negligible, the ZCR response will be received by nodes 2,1, and 0 at $(d_{01}+d_{02})$, $2d_{01}$, and $(2d_{01}+d_{03})$ respectively. Upon receiving the ZCR, response each node then estimates its distance from the parent ZCR using the following formula

$$\begin{aligned} dist_{toparentZCR} = {} & dist_{tolocalZCR} + \\ & (t_{ZCRreplyrecv} - t_{ZCRchallengerecv}) - \\ & dist_{fromlocalZCRtoparentZCR} \end{aligned}$$

The application of this formula yields $d_{02}$, $d_{01}$, and $d_{03}$ for nodes 2, 1, and 3, respectively. At this stage, node 2 will note that it is closer to the parent ZCR than node 1, the current ZCR, and immediately transmits two ZCR takeover packets containing its calculated distance to the parent ZCR. The first packet sent to the child zone informs the other nodes that it is closer to the parent ZCR than node 1, while the second sent to the parent zone informs the parent ZCR that a new representative has been elected for the child zone. Node 3, on the other hand, determines that it is farther away and remains silent. When nodes receive a takeover message from a new ZCR, they replace the identity of any entries that match the old ZCR with the new one. Receivers in the parent zone will not know the distance between themselves and the new ZCR for the child zone. However, as the RTT between the old child ZCR and the parent ZCR is greater than for the new child ZCR and the parent ZCR, using the old RTT value until it is updated does not adversely affect the back-off timers or cause extra retransmissions. One further notes that the application of this formula to receivers 1, 4, and 5 in the fork example shown in the right of Figure 9 also yields the true RTTs, $d_{01}$, $d_{04}$, and $d_{05}$ respectively, and therefore the logic for the star case is the same as for the chain case.

This process of transmitting ZCR challenges is performed periodically by each ZCR, with the time between successive challenges being randomized to prevent synchronization between ZCRs in sibling zones. Since a ZCR might expire, nodes within a child zone maintain their own ZCR timers, but set them so that their firing window is always slightly larger than that of their ZCR. Thus, a non-ZCR will only issue a challenge to the parent in the event that it fails to hear from the local ZCR. The issuance of a ZCR challenge by a non-ZCR node does not automatically result in that node becoming the ZCR, since the old ZCR will still determine that it is closer than the usurper and reassert its superiority as soon as the usurper attempts to issue a takeover message. Furthermore, since ZCR takeover messages contain the new ZCR's estimate of its distance to the parent ZCR, other potential ZCRs should perform suppression as appropriate. The challenge process always results in the closest receiver in the zone being elected as the ZCR.

# 6. Simulations

## 6.1 Session Maintenance Simulations

To prove that SHARQFEC's hierarchical session management scheme allows RTT estimation to be performed in a piecemeal fashion, it was simulated using the UCB/LBNL/VINT network simulator, *ns* [18], and network animator, *nam* [19], packages. Restricted by local memory requirements to simulating networks of only a few hundred receivers, simulations were run using the hybrid mesh tree topology shown in Figure 10. The nodes within this network were arranged such that the sender or top ZCR, node 0, fed data to a 3 level hierarchy of 112 receivers arranged as a mesh of 7 receivers that each fed balanced trees. The links connecting the source to the top 7 receivers in each tree were initialized to 45Mbit/sec with all other remaining links set to a rate of 10Mbit/sec. Latencies between the receivers located within each tree were set to 20ms for each link while the latencies used for the backbone links are shown in Figure 10. (The link loss rates shown do not apply for session traffic).

Other networks that were purely chain- or tree-based were also simulated, and, as expected, the appropriate receivers were elected as the ZCR for each zone with each election at each zone taking either one or two challenges. The application of the dynamic ZCR election process in the network shown in Figure 10 also resulted in the appropriate receiver being elected as the ZCR for each zone. At this stage, tests were run to prove that it was possible for receivers to accurately determine the RTT between themselves and the sender of a packet carrying the necessary partial RTTs between ZCRs at different scopes.

The first part of the test involved choosing a receiver at random from each level in the hierarchy and having it send a fake NACK to the largest scope so that it would be heard by every other receiver. If the receiver could not determine an estimate to the sender, the

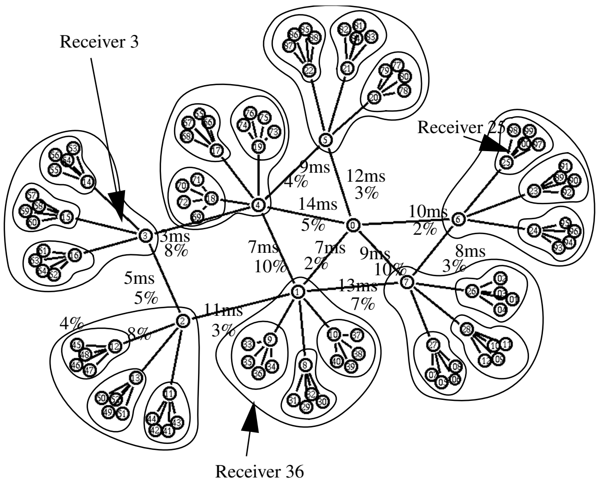


Figure 10. **Test Network used to simulate SHARQFEC within *ns* [18]**

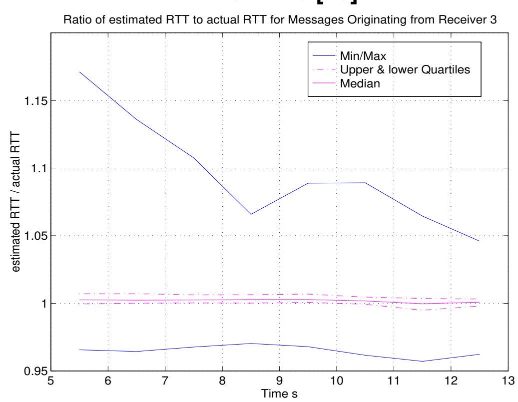


**Figure 11. Ratio of estimated RTTs to actual RTTs for Messages Originating from Receiver 3**

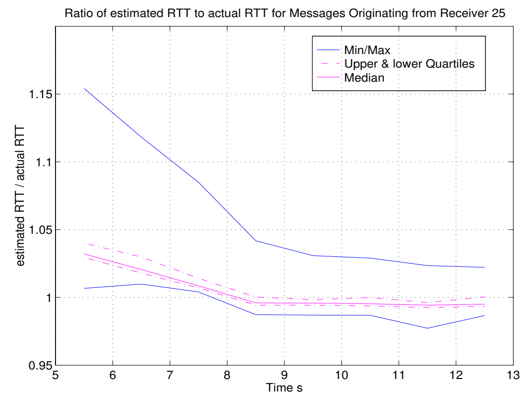


**Figure 12. Ratio of estimated RTTs to actual RTTs for Messages Originating from Receiver 25**

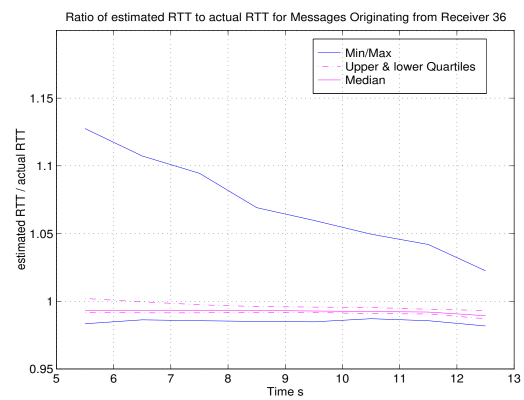


**Figure 13. Ratio of estimated RTTs to actual RTTs for Messages Originating from Receiver 36**

scheme was flawed. The second part involved the selectedreceivers sending NACKs at regular predetermined times. Other receivers that heard these NACKs noted the time of reception, thereby enabling them to determine the actual RTT to the receiver that sent the NACK. The reason for sending multiple NACKs was twofold: first, to prove that estimates were stable; and second, to show that any inaccuracies introduced by a suboptimal ZCR being elected during initialization diminished over time as successive local RTT measurements were made. The results of this test for messages originating from receivers 3, 25, and 36, are shown in Figures 11, 12, and 13. These three figures show that more than 50% of receivers were able to estimate the RTT to a NACK's sender to within a few percent. These figures also show that even when a receiver other than closest one within a zone is initially chosen, the estimates improve over time with each successive measurement. This improvement occurs asymptotically since new measurements are merged with the old using an exponential weighted moving average filter.

## 6.2 Data/Repair Traffic Simulations

Having verified that SHARQFEC's session-maintenance strategy worked, simulations were run to qualify SHARQFEC's performance in delivering data against other delivery schemes [9]. SHARQFEC was compared with an ARQ protocol, and several hybrid ARQ/FEC protocols. SRM was chosen as the ARQ protocol, and its simulation was performed with adaptive timers turned on for best possible performance. The various hybrid ARQ/FEC protocols were simulated by turning off various features within SHARQFEC; scoping (no hierarchy), preemptive FEC packet injection, and sender only repairs. Figures 14 through 21 are annotated with ns = no scoping, ni = no injection, and so = sender only. It should be noted that the SHARQFEC(ns,ni,so) protocol is nearly identical to ECSRM [4] except that ECSRM uses fixed timer windows for its suppression timers, while SHARQFEC(ns,ni,so)'s timer windows are based on the RTTs measured between receivers.

Simulations were performed on the same topology used for simulating the viability of session maintenance strategy. To stress the protocols as realistically as possible, the loss rates for the links were set to ensure that every link suffered some loss, and that some parts of the network suffered greater losses than others. The losses for the mesh part of the topology are included in Figure 10. The loss rate between each of the seven mesh nodes and their three children was set to 8%, while the loss rate between the three children and their children was set to 4%. Thus, when the resulting routing trees were taken into account to the outermost receivers, receivers 53 through 62 experienced the worst loss (on the order of 28.3%) while receivers 89 through 100 experienced the least loss (on the order of 13.4%). Realism was further enhanced by subjecting repair packets to the same loss patterns. Session traffic

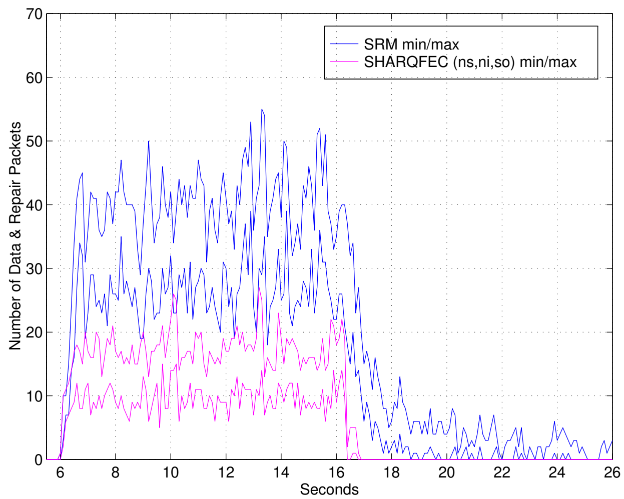


**Figure 14. Data and Repair Traffic - SRM and SHARQFEC(ns,ni,so) / ECSRM**

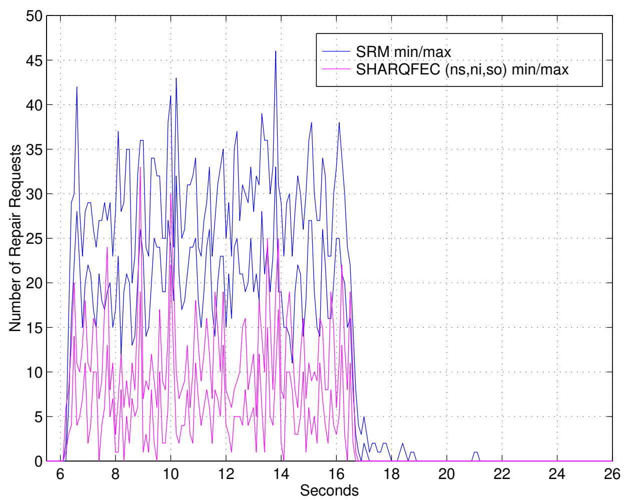


**Figure 15. NACK Traffic - SRM and SHARQFEC(ns,ni,so) / ECSRM**

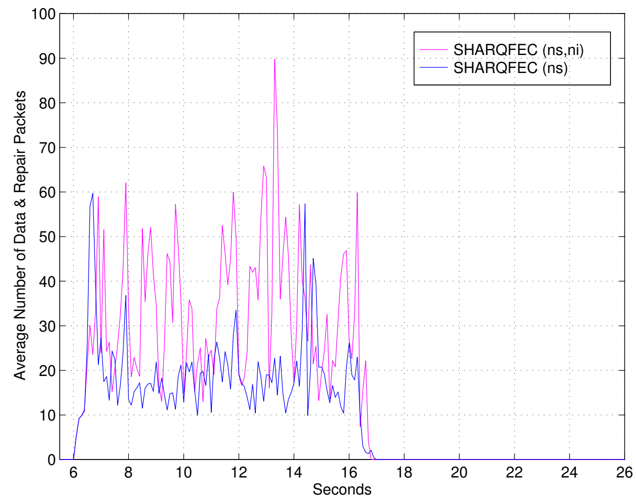


**Figure 16. Average Data and Repair Traffic SHARQFEC(ns,ni) and SHARQFEC(ns)**

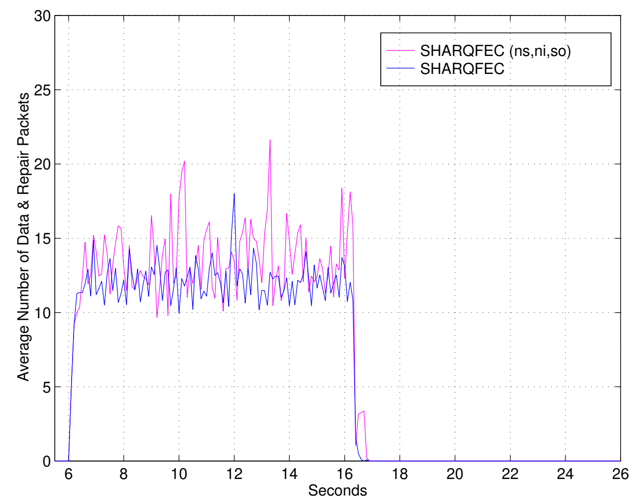


**Figure 17. Average DATA and Repair Traffic - SHARQFEC(ns,ni,so) and SHARQFEC**

and NACKs were not subject to losses.Simulations for each of the protocols were then performed. Each simulation consisted of the nodes joining the session at t = 1 seconds, at which time they began sending session messages to each other. After allowing five seconds for the session state to be established and stabilized, a constant bit rate source at node 0 was turned on at t = 6 seconds. The source then emitted 1024 thousand-byte data packets at a rate of 800Kbit/sec, turning itself off at t = 16.24 sec. The length of the run was chosen to be long enough so that any dependency upon *ns*'s internal random number generator would be minimized when the results were viewed as a whole. Packets were sent in groups of 16 for each of the SHARQFEC runs.

The recovery mechanisms of the SHARQFEC protocol work on groups of packets, so performance for both SRM and SHARQFEC simulations was measured by comparing the sum of data and repair traffic visible at each session over 0.1 second intervals. Thus, if transmission were lossless, one would expect to observe the arrival of ten data packets per measurement interval.

Figures 14 and 15 show that hybrid ARQ/FEC with no scoping, no injection, and repairs sent by the sender only (ECSRM [4]) is able to provide better suppression than SRM, the pure ARQ protocol. One also notes that since repairs are lost as well as data packets and repair timers back-off exponentially, a significant repair tail results. This tail is caused by receivers that send repairs after having missed ones sent by others.

Figure 16 confirms this hypothesis as it shows that ability to suppress decreases for the SHARQFEC (no scoping, no injection) protocol when other receivers are allowed to transmit repairs. It also shows that turning on preemptive repair injection at the source improves suppression, however not to the point where performance is better than the original SHARQFEC(ns,ni,so) /ECSRM protocol shown in Figure 14.

Figure 17 shows that final addition of scoping achieves the desired result of improved suppression. One notes that the average total data plus repair bandwidth seen by the receivers is much better behaved with the peaks at t = 10.1, 13.3, 15.9, and 16.2 seconds

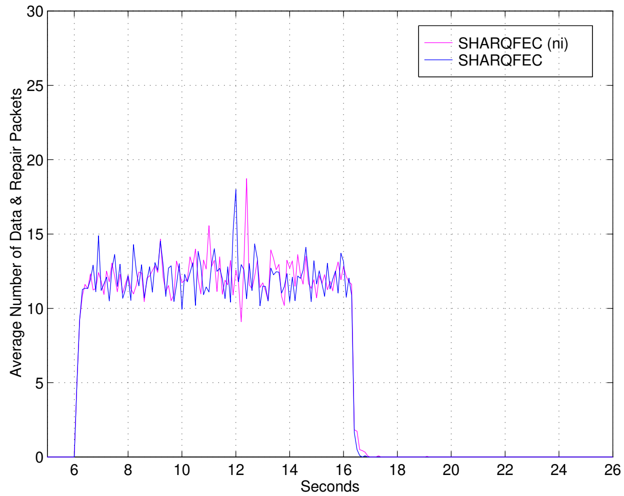


**Figure 18. Data and Repair Traffic SHARQFEC(ni) and SHARQFEC**

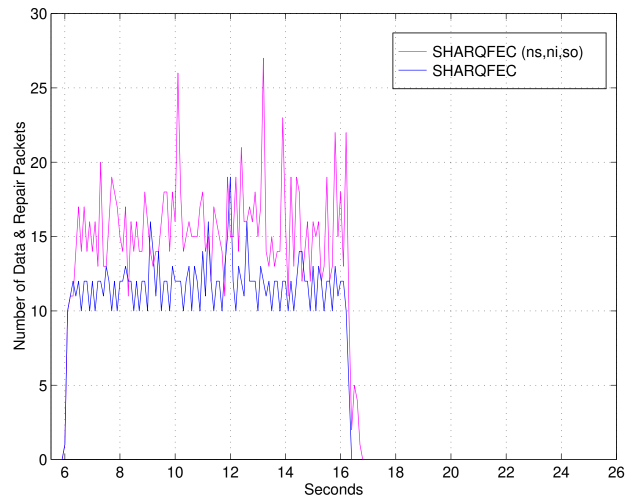


**Figure 20. Data And Repair Traffic seen by the Source for SHARQFEC(ns,ni,so) and SHARQFEC**

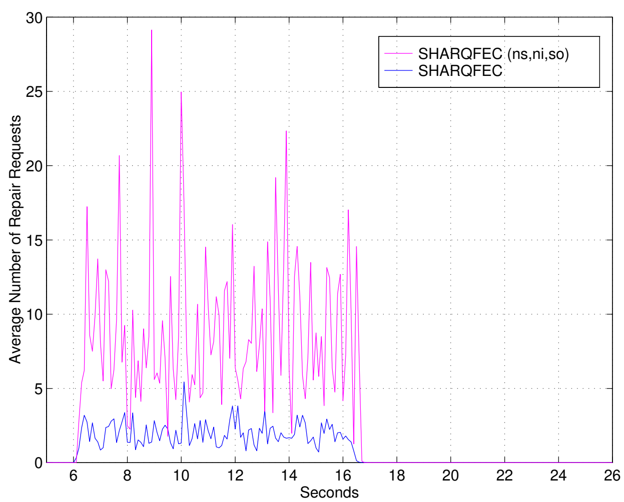


**Figure 19. Average NACK traffic SHARQFEC(ns,ni,so) and SHARQFEC**

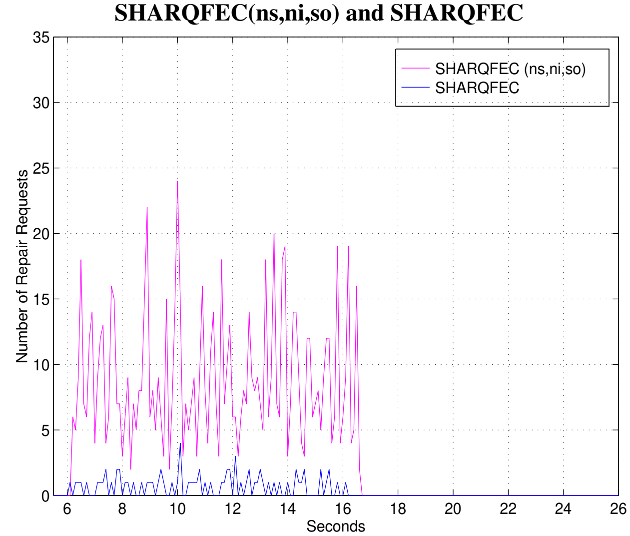


**Figure 21. NACK Traffic seen by the Source for SHARQFEC(ns,ni,so) and SHARQFEC)**

all reduced significantly.

Figure 18 further confirms the result presented by Rubenstein *et al*. in [16] that preemptive or proactive FEC injection does not increase bandwidth. It also further validates that preemptive FEC injection does not cause additional bandwidth within the hierarchical environment used by SHARQFEC. Figure 19 shows that the combinations of hierarchy, and injection affords significantly better suppression of NACKs over the SHARQFEC(ns,ni,so)/ ECSRM model. Comparison with Figure 15 shows that the average number of NACKs seen by each receiver is less than or equal to the minimum seen for the SHARQFEC(ns,ni,so)/ECSRM protocol.

Finally, Figures 20 and 21 show the traffic observed by the source in the core of the network.One notes that the volume of additional traffic above the original transmissions is minimal, and that SHARQFEC's hierarchy is able to reduce backbone traffic considerably by localizing repairs within the smaller administratively-scoped regions.

# 7. Future Work

SHARQFEC currently uses fixed timers for suppression purposes. As was noted in [3] fixed timers are incapable of coping with all network topologies, and therefore inclusion of some mechanism for adjusting the timer constants can lead to enhanced performance. Further work is needed to explore mechanisms for adjusting the timer constants used by SHARQFEC. This task is complicated by the fact that SHARQFEC included mechanisms that preemptively inject repairs into each zone prior to requests being made.

Another avenue for future work would be to explore ways of merging SHARQFEC's functionality with the Real Time Transport Protocol (RTP) [17]. RTP's initial design assumed that by virtue of being a real time protocol, no time would be available to effect repairs, and therefore repair mechanisms need not be included. However, this view has changed somewhat with much work being done to add reliability to RTP in the Audio Visual Transport (avt) group of the Internet Engineering Task Force (IETF) and the

Reliable Multicast (rm) Research Group of the Internet Research Task Force (IRTF). One key area where SHARQFEC may assist in this task would be in solving the RTCP announcement problem. SHARQFEC's hierarchical session management and repair mechanisms could easily be modified to include summaries of Receiver Report (RR) information, thereby increasing RTP's scalability significantly.

SHARQFEC adds hierarchy to the tool-box of mechanisms that can be used to minimize multicast repair traffic. The hierarchy it uses localizes individual repair packets to the portions of the network where they are needed. This same hierarchy also provides the means for localizing late-join traffic. Further results and simulations, along with details of how the hierarchy created by SHARQFEC can be used to affect the significantly larger repairs that result from late-joins can be found in [9].

## 8. Concluding Comments

We designed, developed, and simulated a new reliable multicast protocol called SHARQFEC. SHARQFEC differs from previous reliable multicast efforts in that it relies on the judicious construction of a hierarchy of administratively scoped regions to achieve localization. Consequently, SHARQFEC does not require extensive router modifications, or the creation of additional state inside them, in order to work or to deploy on a widespread scale. Simulations in which both data and repair traffic were subject to loss showed that SHARQFEC can provide reliable delivery under conditions of heavy network loss without causing further congestion. Theoretically, there is no reason why SHARQFEC could not scale to hundreds of thousands or millions of receivers.

We also proposed a new scalable session management mechanism within SHARQFEC for the determination of inter session-member transit times that are commonly used for traffic suppression within reliable multicast protocols. The new method determines theses transit times indirectly and was shown to give transit time estimates that were accurate to within a few percent of the actual transit times. We also show that SHARQFEC's session management mechanism reduces both the traffic volume and session state that must be maintained by each session member.

## 9. Acknowledgments

Much thanks goes to the MIT Media Laboratory where this work was undertaken. Special thanks to Mark Banham, Dave Clark, Jim Gemmell, Mark Handley, Henry Holtzman, Andy Lippman, Carl Malamud, and the SIGCOMM98 reviewers who reviewed this paper for comments that helped to clarify and refine the work presented in this paper.

## 10. References

[1] Berners-Lee, T.J., Cailiau, R., and Groff, J.F., "The World Wide Web", Computer Networks and ISDN Systems, Nov 1992, vol.25, (no4-5), pp 454-9.

[2] Deering, S., and Hinden, R., "Internet Protocol, Version 6, Specification", RFC 1883, Xerox PARC, Ipsilon Networks, December 1995.

[3] Floyd, S., Jacobson, V., McCanne, S., Liu, C., and Zhang, L., "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing", Proceedings of ACM SIGCOMM '95, Cambridge U.S.A, pp342-356.

[4] Gemmell, J., "Scalable Reliable Multicast Using Erasure-Correcting Re-Sends," Microsoft Research Technical Report, MSR-TR-97-20, June 30, 1997.

[5] Handley, M., "On Scalable Multimedia Conferencing Systems", Ph.D. Thesis, University of London, 1997.

[6] Holbrook, H.W., Singhal, S.K., and Cheriton, D.R., "Log-based receiver-reliable multicast for distributed interactive simulation," in Proceedings of ACM SIGCOMM '95, October 1995.

[7] Huitema, C., "The case for packet level FEC", Proceedings of IFIP 5th International Workshop on Protocols for High Speed Networks (PfHSN'96)," INRIA, Sophia Antipolis, FRANCE, October 1995, IFIP, Chapman &Hall.

[8] ISO/IEC 138182-2 (MPEG-2), "Generic Coding of Moving Pictures and Associated Audio - Part 2, Video," International Standard, November 1994.

[9] Kermode, R., "Smart Network Caches: Localized Content and Application Negotiated Recovery Mechanisms for Multicast Media Distribution," Ph.D. Thesis, MIT, June 1998. URL: http://www.media.mit.edu/~woja/thesis.html

[10] Nonnenmacher, J., Biersack, E., Towsley, D., "Parity-Based Loss Recovery for Reliable Multicast transmission," Computer Communications Review ACM SIGCOMM, volume 27, number 4, 1997.

[11] Papadopoulos, C., Parulkar, G., and Varghese, G., "An Error Control Scheme for Large-Scale Multicast," submitted to Infocom 98.

[12] Paul, S., Sabnani, K., Lin, J., and Bhattacharyya, S., "Reliable Multicast Transport Protocol (RMTP)", To appear IEEE Journal on Selected Areas in Communications, special issue on Network Support for Multipoint Communication.

[13] Postel, J., "Internet Protocol", RFC 791, September 1981.

[14] Rizzo, L., "Effective Erasure Codes for Reliable Computer Communication Protocols," ACM Computer Communications Review, Vol. 27, n.2, Apr. 97, pp. 24-36.

[15] Rizzo L., and Vicisano, L., "A Reliable Multicast data Distribution Protocol based on software FEC techniques," Proceedings of the Fourth IEEE, HPCS'97 Workshop, Chalkidiki, Greece, June 1997.

[16] Rubenstein, D., Kurose, J., Towsley, D., "Real-Time Reliable Multicast Using Proactive Forward Error Correction," NOSSDAV '98, Cambridge, UK, July 1998.

[17] Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V., "RTP: A Transport Protocol for Real-Time Applications," RFC 1889, January 1996.

[18] UCB/LBNL/VINT Network Simulator, ns, URL: http://www-mash.cs.berkeley.edu/ns

[19] UCB/LBNL/VINT Network Animator, nam, URL: http://www-mash.cs.berkeley.edu/nam

[20] Yajnik, M., Kurose, J., and Towsley, D., "Packet Loss Correlation in the MBone multicast network," Proceedings of IEEE Global Internet Mini-Conf. GLOBECOM'96, Nov. 1996.

[21] Yavatkar, R., Griffioen, J., and Sudan, M., "A Reliable Dissemination Protocol for Interactive Collaborations," Proceedings of ACM Multimedia 95. 1995.