Activating and Deactivating Repair Servers in Active Multicast Trees[1]

Ping Ji, Jim Kurose, Don Towsley
Department of Computer Science
University of Massachusetts/Amherst
{jiping,kurose,towsley}@cs.umass.edu

Abstract. For time-constrained applications, repair-server-based active local recovery approaches can be valuable in providing low-latency reliable multicast service. However, an active multicast repair service consumes resources at the repair servers in the multicast tree. A scheme was thus presented in [10] to *dynamically* activate/deactivate repair servers with the goal of using as few system resources (repair servers) as possible, while at the same time improving application-level performance. In this paper, we develop stochastic models to study the distribution of repair delay both with and without a repair server in a simple multicast tree. From these models, we observe that the application deadline, downstream link loss rates, the number of receivers, and the upstream round trip time of a repair server all influence the overall value of activating an active repair server. Based on these observations, we propose a modified dynamic repair server activation algorithm that considers the packet loss rate, the number of downstream receivers, and the round trip time to the nearest upstream active repair server when activating/deactivating a repair server. From simulation, we observe that our modified dynamic repair server activation algorithm provides a significant reduction in the latency of successful packet delivery (over the original algorithm) while using the same amount of system resources. We also find that much of the performance gains achievable by having active repair servers can be obtained by having only a relatively small fraction of repair servers actually being active.

1 Introduction

Delay-sensitive applications such as teleconferencing, distributed simulation, multiplayer games, and Internet telephony all have timing constraints on the successful delivery of data from source to destination(s). In such applications, data that do not arrive at receivers prior to some application-determined deadline, are considered lost and can result in impairments in application-level performance. For a real-time video stream, for example, a missed deadline can result in playout jitter or breaks in the playout stream. For a conversation with interaction among multiple speakers, the delay from when a user speaks or moves until the action is manifested at the receiving hosts should be less than a few hundred milliseconds [6].

Many reliable multicast protocols exploit local recovery [2] [13] [7] [14] [11] to reduce the delay in successful data delivery, making them attractive for supporting delay-sensitive applications. Similarly, by using active repair servers (RS) in a multicast tree to provide retransmission (i.e., error recovery) service, repair-server-based local recovery schemes can successfully reduce the repair latency, as well as suppress NAK implosion, and provide retransmission scoping [4]. On the other hand, active repair servers inside

[1] This work is supported by the Defense Advanced Research Projects Agency (DARPA) under contract N66001-9117-411V

the network require additional resources (e.g., buffering and processing). Thus it is desirable to activate as few repair servers as possible, while at the same time providing enough repair servers to improve repair latency.

In this paper, we study the tradeoff between repair delay (equivalently, the time needed to successfully deliver data to the receiver(s)) and system resource consumption by focusing on the benefit gained by using server-based active error recovery (AER) [4]. We begin by developing stochastic models to study the distribution of repair delay both in the presence, and in the absence, of repair servers. Based on these models we observe that the application deadline, downstream link loss rates, the number of receivers, and the upstream round trip time of a repair server are all important criteria influencing the decision of whether to activate/deactivate an RS.

We then consider specific algorithms for dynamic RS activation/deactivation. We begin with the algorithm from [10], which introduced a protocol to dynamically activate/deactivate repair servers on the basis of the packet loss rate within the RS's subtree. For delay-sensitive applications, it is also appropriate to consider time-based measures such as the upstream RTT in deciding whether or not to activate an RS. We thus modify the algorithm in [10] to consider not only the packet loss rate and number of downstream receivers, but also the round trip time to the nearest upstream active repair server (or the sender) when making the activation/deactivation decision. We study via simulation the tradeoff that exists between the fraction of RSs that are activated and the repair delay. We show that our modified dynamic RS activation algorithm provides a significant reduction in repair delay over the original algorithm [10], while using no more system resources than the original algorithm. We also find that much of the performance gains achievable by having active repair servers can be obtained by having only a relatively small fraction of repair servers actually being active.

The remainder of this paper is organized as follows. In Section 2, we describe a multicast loss recovery architecture and a reliable multicast protocol that uses active repair services. Section 3 presents the analytic model that we use to evaluate the decrease in repair delay when using an active repair server for time-constrained applications in a simple multicast tree. Section 4 proposes a modified algorithm for dynamically activating/deactivating repair servers, and presents simulation results comparing the performance of the modified algorithm with that of the original algorithm. Section 5 concludes this paper.

2 Real-time Reliable Active Multicast: Motivation and Algorithms

Several recent efforts have focused on providing *better-than-best-effort* service for delay-sensitive applications. Maxemchuk et al. [9], Lucas et al.[8] and other researchers have proposed various distributed *receiver*-based local recovery schemes. The Active Error Recovery (AER) protocol [4] uses a *repair-server*-based local recovery scheme, in which a repair server (RS) is attached to a router to perform active local error recovery (i.e., buffering and retransmission) for downsteam receivers and RSs. AER was shown to achieve low latency error recovery, NAK suppression, retransmission scoping, and superior bandwidth utilization. However, the use of additional active nodes inside the network comes at a price - additional system resources, such as buffers and

computation, are needed at the active nodes. Several questions immediately arise - how many active RSs are needed and where should they best be placed; what is the trade-off between the number of activated RSs and the repair latency; must a repair server always be active, or can a repair server dynamically monitor performance and then self-activate/deactivate according to observed performance. These are some of the questions we address in this paper.

Rubenstein et al.[12] have proposed a *static* centralized tree-based protocol to construct a *repair graph* that uses RSs to retransmit lost packets to receivers before their deadline. However, when the multicast tree structure and/or link loss rates change over time, such a static approach may not be appropriate. In [10], Osland et al. presented an algorithm to *dynamically* activate/deactivate RSs in response to changing network conditions. However, their approach uses only the packet loss rate to trigger RS activation/deactivation. We will see shortly that for delay-sensitive applications, delay-based criteria can be effectively used in making dynamic activation/deactivation decisions at an RS.

Let us now describe a reliable multicast protocol known as AER (Active Error Recovery [4] [5] [10] [1]) that implements active server-based repair services; we will subsequently modify this protocol to implement dynamic RS activation/deactivation. In AER, *subcast* is defined to be a multicast transmission from an RS to the downstream multicast subtree rooted at the RS. We describe the algorithm in the context of a sender that periodically multicasts data to a multicast address that is subscribed to by all receivers and participating Repair Servers; other scenarios are also possible. The algorithm operates as follows:

- **Packet forwarding, buffering.** When a new packet arrives at a router associated with an RS, it is multicast downstream by the router and stored in the buffer of the repair server.
- **NAK suppression.** On detecting a loss, a receiver (or repair server), after waiting for a random period of time (the NAK suppression time), sends out a NAK to its nearest upstream active server (a repair server or the sender). At the same time, it starts a NAK retransmission timer. If the receiver (or repair server) receives a NAK for the same packet from its upstream repair server prior to sending its own NAK, it suppresses its own NAK transmission.
- **NAK timeout.** The expiration of the NAK retransmission timer at a repair server (or a receiver), without prior reception of the corresponding repair, serves as the detection of a lost packet for the repair server (or the receiver) and a NAK is retransmitted.
- **Repair packet transmission, NAK aggregation and propagation.** When a repair server receives a NAK from a downstream node, it subcasts the packet if it has the packet in its buffer. Otherwise, if there is already a pending NAK for that lost packet, the new incoming NAK is suppressed. If there is neither a pending NAK for this packet, nor a buffered repair packet, the RS *immediately* subcasts the NAK downstream and sends a NAK upstream after waiting for a random period of time, as well as keeping a pending NAK until the repair packet arrives.
- **Repair packet retransmission.** On receiving a NAK from a downstream repair server, the sender re-subcasts the requested packet to all the receivers and repair

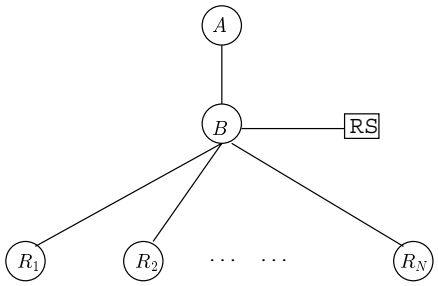

Fig. 1. A Simple Multicast Topology

servers. As mentioned above, these repairs are received by intermediate repair servers and forwarded down the tree only if there is a pending NAK for that repair.

– **Repair packet reception.** If a repair packet is received, the repair server first checks whether there is a pending NAK for that packet. If so, that repair packet will be buffered and subcast downstream. If there is no pending NAK for this repair packet, the RS discards this packet.

The AER protocol successfully reduces NAK implosion by using randomized timer-based NAK suppression and by using the repair server hierarchy for NAK aggregation. We next present a simple model of the delay performance of this protocol in a simple multicast tree.

3 A Simple Model for Understanding the Benefits of Using A Repair Server in Time-Constrained Applications

In this section we develop an analytic model to study the distribution of repair delay both with and without a repair server for time-constrained applications. Our goal is to gain insight into the effect of parameters such as application deadline, downstream link loss rates, the number of receivers, and the upstream round trip time of a repair server.

Figure 1 shows a generic model for a single RS co-located at a router. Node A is the sender. Node B represents an intermediate router, to which a repair server (that is denoted as a square node in this graph) can be attached. On receiving a packet from sender A, router B multicasts that packet on its subtree. $R = \{R_1, R_2, ..., R_N\}$ denotes the set of receivers on the tree. The solid line between node A and node B represents the transmission path between the sender and the router (or its corresponding repair server). Similarly, the lines between router B and receivers R_1 to R_N represent the paths between the router and receivers.

The notation we will use is as follows:

X_i: Time required to successfully deliver a packet from the sender to receiver R_i.

X_{ij}: Time to transmit a packet from i to j in the absence of packet loss, where $i \in \{A, B\}$ and $j \in \{\{B\} \cup R\}$. We model X_{ij} as a fixed value.

X_{ji}: Time to transmit a NAK from j to i, where $j \in \{\{B\} \cup R\}$ and $i \in \{A, B\}$. We model X_{ji} as a fixed value.

τ_i: A joint timer of i, that integrates the NAK suppression timer, which is a function of the RTT between i and its nearest upstream active node (the sender or the *active* RS), and the NAK retransmission timer of i, which is a function of the RTT from i to the sender. Here $i \in \{\{B\} \cup R\}$.

D: The application-dependent deadline

p_{ij}: Loss probability of path ij, where $i \in \{A, B\}$ and $j \in \{\{B\} \cup R\}$.

λ: Recall that we are modeling a constant rate sender. Data packets arrive at A for first-tie transmission at constant rate λ.

We denote the probability that a packet is delivered after the application deadline at receiver R_i as $P\{X_i > D\}$.

Based on this single RS configuration, we now describe two simple models that characterize the distribution of repair delay both with and without RSs.

3.1 Evaluation of $\sum_{i \in R} P\{X_i > D\}$ When an RS is Used

Let us first consider the transmission delay, X_i, in the presence of a repair server (Figure 1). We introduce the following random variables. Let K denote the number of losses prior to the first successful transmission of a packet on path AB. Let K_i denote the number of losses prior to the first successful transmission of a packet on path BR_i. We are interested in the value of X_i conditioned on $K = k$ and $K_i = k_i$. We denote this as $X_i(k, k_i)$ and compute its value as follows:

$$X_i(k,k_i) = \begin{cases} X_{AB} + X_{Bi} & k = 0, k_i = 0 \\ X_{AB} + \Delta_i + (k_i - 1)\tau_i + X_{iB} + X_{Bi} & k = 0, k_i \geq 1 \\ \Delta_{RS} + (k-1)\tau_B + X_{BA} + X_{AB} + X_{Bi} & k \geq 1, k_i = 0 \\ \Delta_{RS} + (k-1)\tau_B + X_{BA} + X_{AB} + \Delta + & \\ (k_i - 1)\tau_i + X_{iB} + X_{Bi} & k \geq 1, k_i \geq 1 \end{cases} \quad (1)$$

A timeline of X_i, when $k \geq 1$ and $k_i \geq 1$, is shown in Fig. 2. The loss detection duration, Δ_{RS}, is a function of the constant transmission rate λ and the delay of path AB. At the repair server, the time until the next successful receipt of a packet follows a geometric distribution with mean $\frac{1}{1-p_{AB}} \cdot \frac{1}{\lambda}$. Thus we model $\Delta_{RS} = X_{AB} + \frac{1}{1-p_{AB}} \cdot \frac{1}{\lambda} + \tau_B^{'}$, where $\tau_B^{'}$ is the NAK suppression timer of repair server B. Similarly we model $\Delta_i = X_{Bi} + \frac{1}{1-p_{Bi}} \cdot \frac{1}{\lambda} + \tau_i^{'}$, where $\tau_i^{'}$ is the NAK suppression timer of receiver R_i. Here, Δ denotes the time between the arrival of a retransmitted packet at the repair server and the receiver's transmission of the next NAK. From Figure 2, we observe that Δ follows a uniform distribution in $[0, \tau_i]$.

For a given k and a given k_i, from equation (1), we can easily evaluate the value of $X_i(k, k_i)$. Consequently, we can determine the probability that $X_i(k, k_i)$ is greater than the application deadline D. The probability $P\{X_i > D\}$ can be evaluated by removing the conditioning on the values of k and k_i:

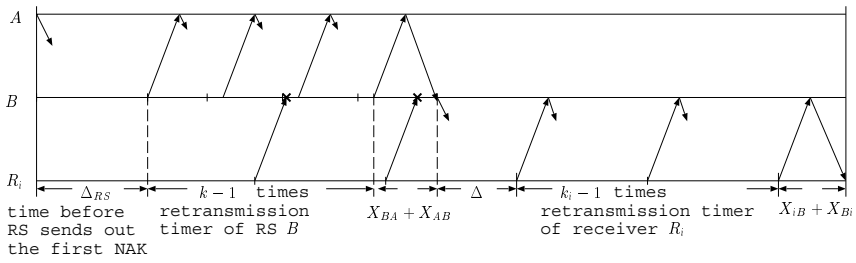

Fig. 2. Timeline of $X_i(k, k_i)$, Repair Service is Used

$$\sum_{i \in R} P\{X_i > D\} = \sum_{k=0}^{\infty} (\sum_{i \in R} \sum_{k_i=0}^{\infty} \mathbf{1}(X_i(k, k_i) > D) \cdot P\{K_i = k_i\}) \cdot P\{K = k\} \quad (2)$$

where, $P\{K_i = k_i\} = p_{Bi}^{k_i} \cdot (1 - p_{Bi})$, $P\{K = k\} = p_{AB}^{k} \cdot (1 - p_{AB})$ and $\mathbf{1}(P)$ is one when the predicate P is true and zero otherwise.

3.2 Evaluation of $\sum_{i \in R} P\{X_i > D\}$ When RS is Not Used

The analysis is more complicated when there is no repair server attached at router B of Figure 1, because the end-to-end link losses are correlated. Thus, the probability that a packet is delivered after the application deadline cannot be solved independently at each receiver. In this case, we can use the following alternative approach to evaluate $\sum_{i \in R} P\{X_i > D\}$. Again, we focus on a randomly selected packet.

$$\begin{aligned} \sum_{i \in R} P\{X_i > D\} &= E[\text{number of receivers that receive the packet after the deadline}] \\ &= N - E[\text{number of receivers that receive the packet by the deadline}] \end{aligned} \quad (3)$$

where N denotes the number of receivers. To evaluate the expected number of receivers that successfully receive the data packet before its deadline, we must define some additional notation.

Let $T^{(l)}$ be the number of receivers that successfully receive that data packet before or at the l-th retransmission of that packet. Thus, from formula (3), we can derive

$$\sum_{i \in R} P\{X_i > D\} = N - E[T^{(d)}] \quad (4)$$

where d is the maximum number of retransmissions allowed to meet the deadline. Here d is a function of the application-dependent deadline, D, and link transmission delays. Yamamoto et al. [15] provide a general model for evaluating the delay performance of receiver-initiated reliable multicast protocols. In our study, we assume that a NAK-receipt suppression timer is used by the sender. Within a NAK-receipt timeout interval,

the sender considers multiple NAKs it receives for a given packet as being redundant and only retransmits (multicasts) the packet once. This timer is a function of the longest RTT to receivers. We simplify our problem to the case that all receivers have the same RTT to the sender. Therefore, at receiver R_i, during each timer τ_i, there can be at most one retransmission for a specific lost packet. Thus we have $d = \lfloor \frac{D-\Delta_i-X_{iA}-X_{Ai}}{\tau_i} + 1 \rfloor$, where Δ_i denotes the loss detection duration of receiver R_i and is a function of p_{Ai}, X_{Ai} and λ. Consequently, from equation (4), we have

$$\sum_{i \in R} P\{X_i > D\} = N - \sum_{i=1}^{N} i \cdot P\{T^{(d)} = i\} \tag{5}$$

Given the application deadline and the corresponding maximum number of retransmissions d, we can calculate the probability $P\{T^{(d)} = i\}$ as follows:

$$P\{T^{(l)} = i\} = \sum_{j=0}^{i} P\{T^{(l)} = i | T^{(l-1)} = j\} \cdot P\{T^{(l-1)} = j\} \qquad l = 1, 2, ..., d \tag{6}$$

where the conditional probability of $P\{T^{(l)} = i | T^{(l-1)} = j\}$ is shown in equation (7).

$$P\{T^{(l)} = i | T^{(l-1)} = j\} = \begin{cases} p_{AB} + (1 - p_{AB}) p_{Bi}^{N-j}, & i = j; \\ (1 - p_{AB}) \binom{N-j}{i-j} p_{Bi}^{N-i} (1 - p_{Bi})^{i-j}, & N > i > j; \end{cases} \tag{7}$$

with the initial condition:

$$P\{T^{(-1)} = 0\} = 1$$

Thus $P\{T^{(d)} = i\}$ can be recursively computed and the corresponding $\sum_{i \in R} P\{X_i > D\}$ can be evaluated.

3.3 Numerical Case Study

Using the analysis techniques described above, we can evaluate the cumulative distribution function of the X_i, the time needed to successfully deliver a packet to a receiver in the presence/absence of a RS for various application-level deadlines, D. A numerical case study for the CDF of X_i is shown in Figure 3. Here, we assume the number of receivers is 5, path AB experiences a loss rate as $p_{AB} = 0.05$. For each downstream path, we assume a loss rate of $p_{Bi} = 0.10$. The one-way delay for path AB is 30ms, and the one-way delay between router B and each receiver R_i is 10ms. Figure 3 shows that when a repair server is active, there is a higher probability that a packet is successfully delivered within a given deadline than the case when the repair server is not active. For this set of parameters, we see the largest gain when the application deadline D is between $80ms$ to $190ms$.

As the values of the model parameters change, so too does the distribution of X_i. Due to space considerations, we only briefly explore the parameter space to identify general trends; in [3] we provide a more complete study. In [3] we define a general cost function that weights the benefits of decreased repair latency and the cost of resources

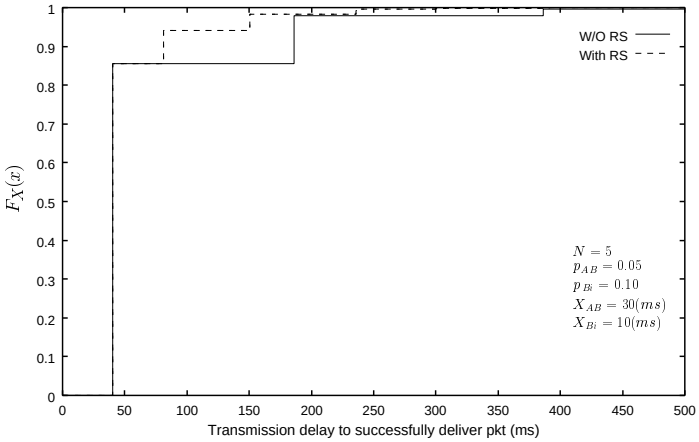

Fig. 3. Compare the CDF of Transmission Delay for With/Without RS Cases

(e.g., buffering and computation) required by active repair servers. The cost difference between the case of having active repair servers and not having active repair servers, is denoted by ΔC, and is defined as:

$$\Delta C = c_a P\{X_i^{noRS} > D\} - (c_a P\{X_i^{withRS} > D\} + c_b)$$

where c_a and c_b are weights reflecting the unit performance degradation penalty cost for an overly delayed packet and the resource cost for recovering a packet, respectively.

We illustrate the effect of an application's deadline on the cost difference ΔC in Figure 4. We considered four downstream loss rates and observed that as the downstream link loss probability p_{Bi} increases, so too does the relative benefit of having active repair servers. For the parameters considered in Figure 4, the benefit is largest when the application deadline lies in the range of 2 to 4.6 times the one-way delay (about $80ms$ to $190ms$). As the application deadline increases beyond this value, the relative performance benefits decrease and actually become negative when $D \geq 380ms$. This is because as the application deadline increases the probability of successfully recovering a packet within the deadline increases (approaching 1) both with and without repair servers, while the with repair server case incurs a resource cost, c_b. In [3], we also consider the effects of downstream loss rates (p_{Bi}), upstream round trip time (X_{AB}) and number of receivers (N) on cost difference ΔC.

4 An Algorithm for Dynamic Activation/Deactivation of Repair Servers

In [10], Osland et al. modified the basic AER protocol to include a two-threshold algorithm for dynamically activating and deactivating repair servers. In their approach, an RS estimates the loss rate to receivers in its subtree over intervals of time. At the end

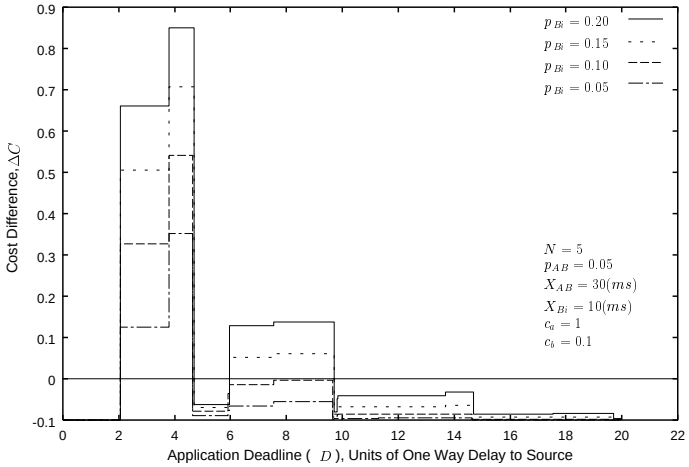

Fig. 4. The Effect of Application Deadline on the Saving of Transmission Delay by Setting a RS

of each interval, if the measured loss rate is greater than an upper threshold value, the repair server is activated; if the measured loss rate is less than a lower threshold value, the corresponding repair server is deactivated.

Our goal is to develop a new algorithm that dynamically activates/deactivates a RS for the purposes of supporting time-constrained applications. In Section 3, we studied how system parameters such as the upstream RTT, number of receivers, application deadline, and link loss rate determine the benefits possible with the use of active RSs. Given that our primary application-level performance metric of interest - the probability that a packet is successfully delivered within its deadline is time-related, it is natural to include time-based considerations into the activation/deactivation decision.

Informally, our time-sensitive RS activation/deactivation algorithm operates as follows. As in [10], an RS estimates (observes) the loss rate to receivers in its subtree over intervals of time of length τ. We introduce variable N_{loss} to denote the number of lost packets during a time interval, and N_{pkt} to denote the number of packets sent by the sender during the same time interval. N_{loss} and N_{pkt} are measured at the RS. Let p_{loss}^k denote the loss probability of an RS's subtree during time interval k. The formula for computing p_{loss}^k is:

$$p_{loss}^k = \frac{N_{loss}}{N_{pkt}} \tag{8}$$

Rather than use the measurement of lost packets to define p_{loss}^k above from [10], we can derive the following expression for p_{loss}^k,

$$p_{loss}^k = 1 - \prod_{i=1}^{n}(1 - p_i) \tag{9}$$

where n is the number of receivers suffering loss and p_i is the link loss rate between the RS and receiver R_i. Our earlier analysis showed that the number of receivers and the individual link loss rates were important factors in determining performance. We note that both of these factors appear explicitly in equation (9).

In addition to using the number of receivers and the individual link loss rates in the activation/deactivation decision, we will want to include time-based measures as well. In our analysis in Section 3, we saw that the upstream RTT (the round trip time from an RS to the closest upstream RS or the sender itself - X_{AB} in Figure 1) was an important factor affecting the delay distribution. Thus, we would like to incorporate the upstream RTT into our dynamic activation/deactivation algorithm. It is worth mentioning here that a crucial parameter affecting performance is the *application deadline* D. However, D is completely application-dependent and can easily be determined only at the receivers. Thus, we choose not to include D in our dynamic repair server activation/deactivation algorithm.

4.1 Algorithm Description

We now modify the RS activation/deactivation algorithm presented in [10] to account for the *upstream RTT* of a RS with p^k_{loss}. Intuitively, if an RS is very close to its upstream RS (or the sender), activating the downstream RS will not significantly reduce the repair delay. On the other hand, if the RTT between an RS and its upstream active repair node is large, activating this RS can result in *local* repair service to downstream nodes (i.e., repairs can be supplied by the RS itself), providing the possibility for a significant reduction in repair delays. This intuition tells us that the larger the upstream RTT of a RS, the more important it is to activate that RS. Therefore, we use the product of the upstream RTT of an RS and its packet loss rate during time interval k as a metric to control the activation/deactivation decision at the end of a time interval.

In the original dynamic RS activation/deactivation algorithm [10] (which we will refer to as AL1), during a time interval (of length $\tau = 4sec$), the number of lost packets and the number of packets sent by the sender are measured by an RS. The packet loss probability of a RS's subtree for time interval k, p^k_{loss}, is then calculated using equation (8). The exact mechanism for estimating the packet loss probability, $\hat{p}^k_{loss}$ is

$$\hat{p}^k_{loss} = (1 - \alpha) \cdot \hat{p}^{k-1}_{loss} + \alpha \cdot p^k_{loss}$$

where α is a smoothing parameter. At the end of each time interval, AL1 compares $\hat{p}^k_{loss}$ with two thresholds. If $\hat{p}^k_{loss}$ is greater than the upper threshold, then the corresponding RS will be active during the next time interval, otherwise if $\hat{p}^k_{loss}$ is less than the lower threshold, the corresponding RS will be inactive during time interval $k + 1$.

AL1 only considers the packet loss probability of a RS's subtree, $\hat{p}^k_{loss}$, as the single metric to control a RS's activation/deactivation decision. In our modified algorithm (which we will refer to as AL2), during each time interval the packet loss rate of a RS's subtree, as well as the RS's upstream RTT, is measured. The product of the upstream RTT of an RS and $\hat{p}^k_{loss}$ of its subtree is then used as the metric for RS activation/deactivation. We denote the product of the upstream RTT and $\hat{p}^k_{loss}$ as ϕ:

$$\phi = R_u \times \hat{p}^k_{loss} \tag{10}$$

where R_u denotes the round trip time from the current repair server to its nearest upstream repair server (or the sender). Dynamic RS activation/deactivation is controlled by a two-threshold mechanism based on the value of ϕ. If ϕ is greater than the modified upper limit, then the corresponding RS will be activated during the subsequent time interval; if ϕ is smaller than the modified lower limit, then the corresponding RS will be deactivated.

4.2 Comparison of AL1 and AL2 via simulation

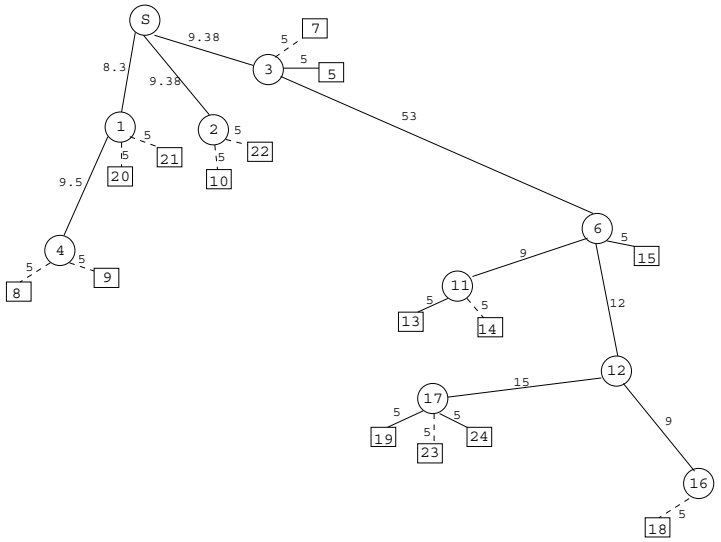

Fig. 5. Simulation Topology 1

We evaluate the new two-threshold policy on three topologies[2]. Figure 5 illustrates the first topology. Focusing on the topology in Figure 5, nodes 1 2 3 4 6 11 12 16 17 are routers attached to repair servers. S represents the sender (source). The remaining nodes are receivers. Link propagation delays are marked in the graph in units of milliseconds. Lossy links are represented by dashed lines. We assume that other links are lossless.

In order to characterize the system-wide extent to which the repair servers are activated, we introduce the active fraction, ρ as follows:

$$\rho = \frac{\text{Total active duration of all RSs}}{\text{Total running time} \times \text{Number of RSs}} \tag{11}$$

[2] The three topologies were originally generated by Diane Kiwior of The Analytic Sciences Corporation, TASC

ρ is taken as a measure of operational cost. The other measure of interest to us is the average repair latency. We focus on the tradeoff between ρ and the average repair latency provided by algorithms AL1 and AL2. This is done through simulation. Here by *average repair latency* we mean the average latency of recovering lost packets at all of the receivers.

Note that by varying the thresholds, the fraction of repair servers that are active will also vary. Figure 6 shows the relationship between active fraction ρ and average repair latency, the curves being generated by changing the value of the thresholds. In our figures, AL1 represents the original algorithm, in which loss probability $\hat{p}^k_{loss}$ was the only metric considered, while AL2 represents the modified algorithm that combines the parameters of upstream round trip time and $\hat{p}^k_{loss}$. We observe from Figure 6 that AL2 can produce a lower average repair latency for the same value of ρ. That is, while consuming the same amount system resources as AL1, AL2 results in the successful delivery of packets with lower average delay.

As a second comparison of interest, Figure 7 plots the *worst average repair latency* - the largest average repair latency experienced over all receivers - for AL1 and AL2. Consistent with the results from Figure 6, the modified algorithm also reduces the worst average repair latency (for the same value of ρ).

Another interesting result illustrated by Figures 6 and 7 is that the average repair latency and worst average repair latency decrease rapidly as the fraction of active repair servers increases from zero (i.e., no active repair servers are ever active) to an active fraction of approximately 10 percent. This indicates that much of the performance gain by having active repair servers can be obtained by having only a relatively small fraction of repair servers being active.

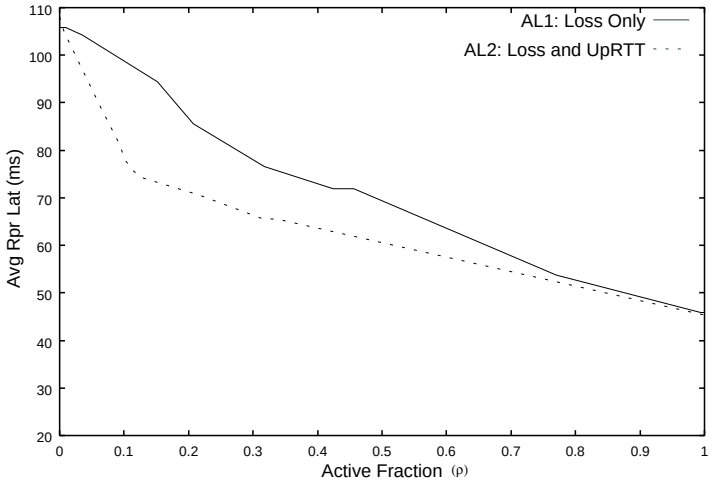

Fig. 6. Average repair latency versus active fraction, topology 1

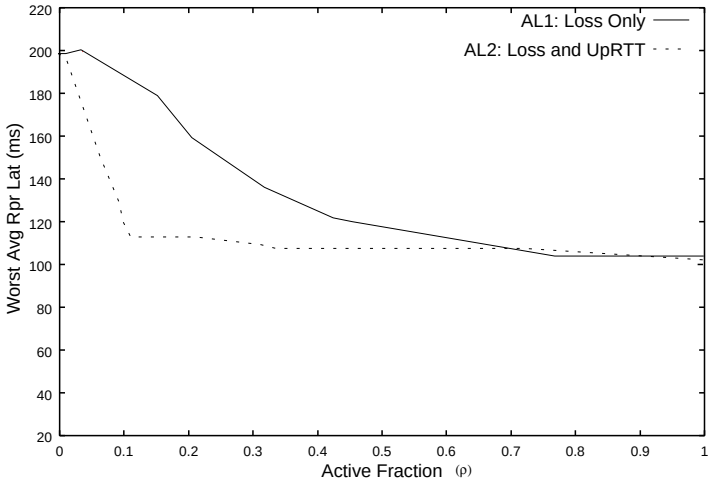

Fig. 7. Worst Average Repair Latency versus active fraction, topology 1

We next introduce a new metric to compare the throughput of AL1 and AL2. The *system repair throughput overhead* counts to total number of link traversals by all retransmitted packets during the simulation. A smaller system repair throughput overhead indicates that less link bandwidth is used in the network to recover from lost packets. We observe, for topology 1, that AL2 results in a smaller repair throughput overhead than AL1, as shown in Figure 8.

Recall that AL2 uses the upstream RTT in determining the activation/deactivation status of an RS. That means, among several subtrees suffering a similar loss rate, the RS with a long-delay link to its parent will be activated first. For a given amount of repair server resource usage, this could possibly result in higher repair traffic for the system as a whole. Figure 9 shows a second topology, for which AL2 reduces (over AL1) both the average repair latency over all receivers and the largest average repair latency, as shown in Figures 10 and 11. However, for this topology, from Figure 12, we notice increased retransmission throughput overhead incurred by using AL2 in comparison to AL1 - the opposite of what we observed in topology 1. This indicates that the retransmission throughput overhead gains in AL1 versus AL2 are topology dependent. A third topology and its corresponding simulation results can be found in [3].

5 Conclusion

In this paper, we have studied the tradeoff between the time needed to successfully deliver data to the receiver(s) and system resource consumption in server-based active error recovery multicast networks. We began by developing stochastic models to study the distribution of repair delay both in the presence, and in the absence, of repair servers.

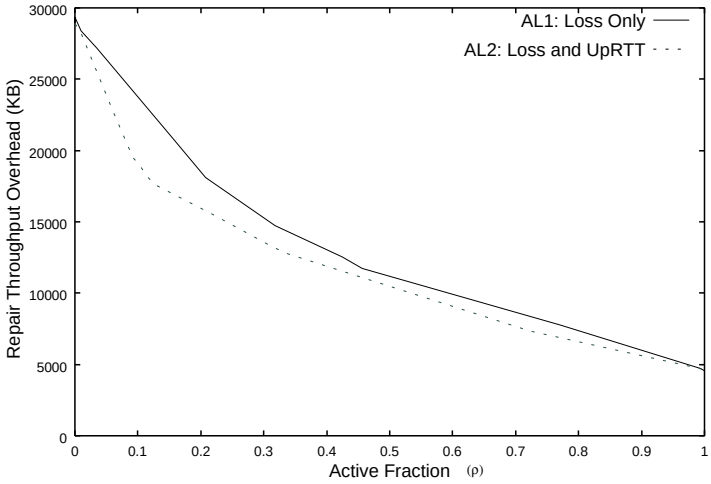

Fig. 8. Repair Throughput Overhead versus active fraction, topology 1

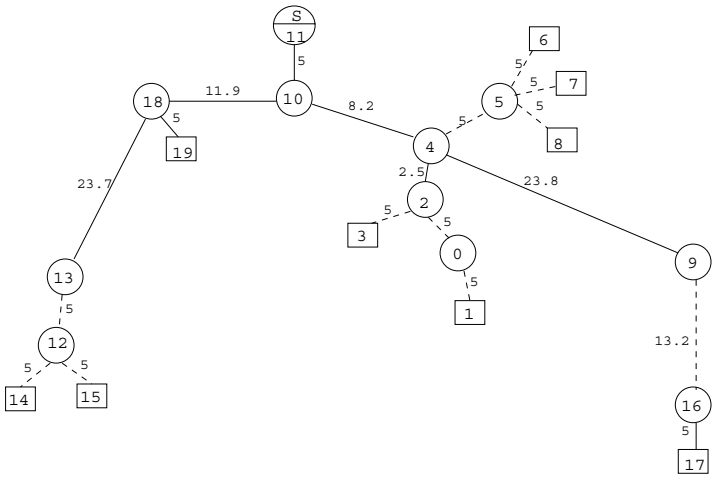

Fig. 9. Topology 2

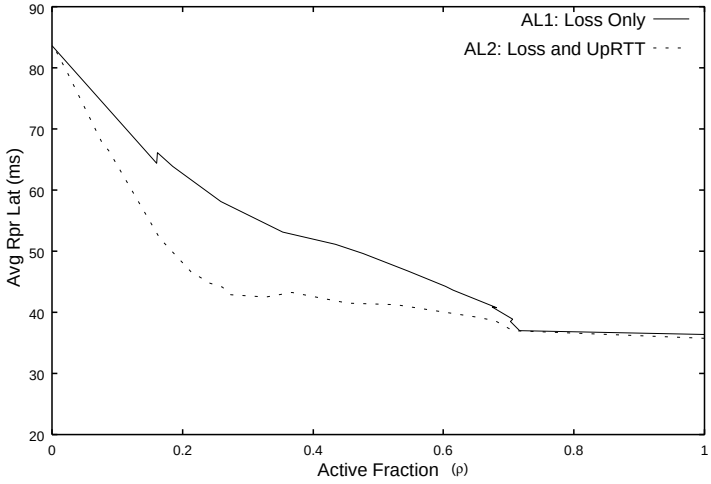

Fig. 10. Average repair latency versus active fraction, topology 2

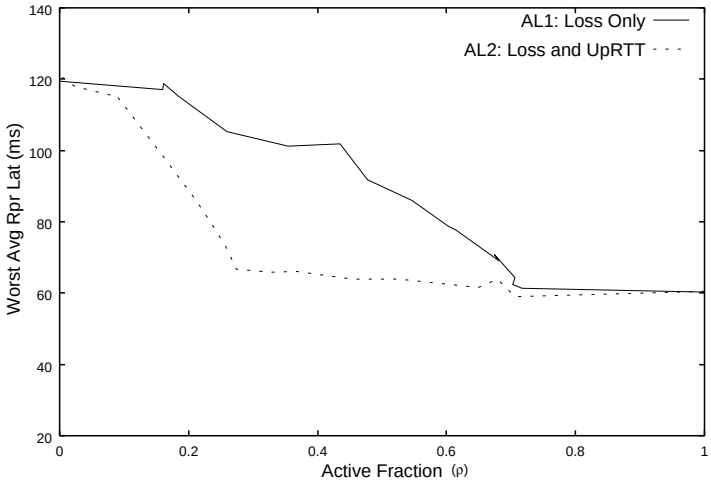

Fig. 11. Worst average repair latency versus active fraction, topology 2

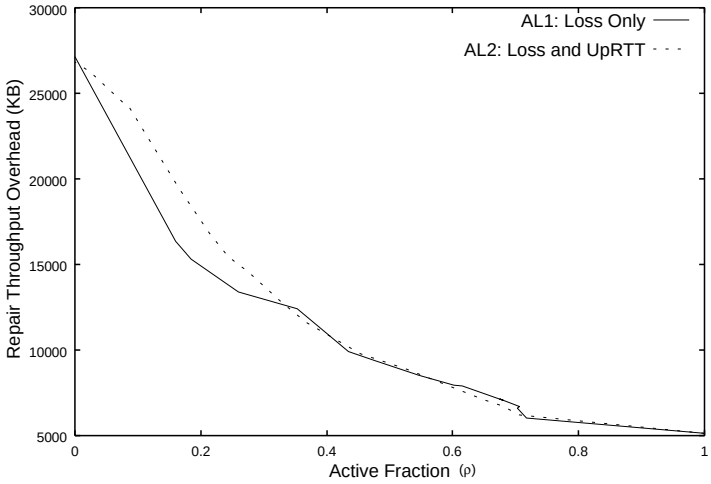

Fig. 12. Repair throughput overhead versus active fraction, topology 2

Based on these models we observed that the application deadline, downstream link loss rates, the number of receivers, and the upstream round trip time of a repair server were all important criteria influencing the decision of whether to activate/deactivate an RS.

Recognizing the value of explicitly consider time-sensitive parameters in determining when an RS should be active, and when it should not, we modified the algorithm in [10] to consider not only the packet loss rate and number of downstream receivers, but also the round trip time to the nearest upstream active repair server (or the sender) when making the activate/deactivate decision. We studied the tradeoff that exists between the fraction of RSs that are activated, the amount of overhead traffic, and the latency of successful packet delivery. We found that our modified dynamic RS activation algorithm provides a significant reduction in repair delay over the original algorithm [10], while using no more system resources than the original algorithm. We also found that much of the performance gains achievable by having active repair servers can be obtained by having only a relatively small fraction of repair servers actually being active.

Our future research in this area will investigate performance under bursty link loss rates. We expect the advantages here (over the case of statically configured repair servers that are always active) to be even more significant, as repair servers can be adaptively activated whenever burst loss occurs. An additional interesting area for study is to develop theoretical models that characterize the performance gains possible with active within-the-network servers, as a function of the density of such servers.

Acknowledgement

The authors would like to thank Zihui Ge for his useful suggestions and Diane Kiwior for her help in simulation.

References

1. http://www.tascnets.com/panama/aer/index.html active error recovery (aer) website.
2. S. Floyd, V. Jacobson, S. McCanne, C. Liu, and Zhang L. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6):784–803, December 1997.
3. Ping Ji, James Kurose, and Don Towsley. Activating and deactivating repair servers in active multicast tree. Technical Report 01-18, University of Massachusetts at Amherst, June 2001.
4. Sneha K. Kasera. *Scalable Reliable Multicast in Wide Area Networks*. PhD thesis, University of Massachusetts at Amherst, 1999.
5. Sneha K. Kasera, Supratik Bhattacharyya, Mark Keaton, Diane Kiwior, Jim Kurose, Dowsley, and Steve Zabele. Scalable fair reliable multicast using active services. IEEE Networks Magazine, 2000.
6. James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*, chapter 6. ADDISON WESLEY, 2000.
7. L. Lehman, S. Garland, and D. Tennenhouse. Active reliable multicast. In *IEEE Infocom*, March 1998.
8. M.T.Lucas, B.J.Dempsey, and A.C.Weaver. Mesh: Distributed error recovery for multimedia streams in wide-area multicast. In *IC3N'97*, 1997.
9. N.F.Maxmchuk, K.Padmanabhan, and S.Lo. A cooperative packet recovery protocol for multicast video. In *ICNP'97*, October 1997.
10. Per-Oddvar Osland, Sneha K. Kasera, Jim Kurose, and Don Towsley. Dynamic activation and deactivation of repair servers in a multicast tree. Technical report of computer science department, University of Massachusetts at Amherst, 1999.
11. C. Papadopoulos, G. Parulkar, and G. Varghese. An error control scheme for large-scale multicast applications. In *IEEE Infocom*, March 1998.
12. Dan Rubenstein, Nicholas F. Maxemchuk, and David Shur. A centralized, tree-based approach to network repair server for multicast streaming media. In *NOSSAV*, 2000.
13. S.Paul, K.K.Sabnani, J.C.Lin, and S.Bhattacharyya. Reliable multicast transport protocol (rmtp). *IEEE Journal on Special Areas in communications*, April 1997.
14. Tony Speakman, Dino Farinacci, , Steven Lin, and Alex Tweedly. Pragmatic general multicast internet draft. August 1998.
15. Miki Yamamoto, Jim Kurose, Don Towsley, and Hiromasa Ikeda. A delay analysis of sender-initiated and receiver-initiated reliable multicast protocols. In *IEEE Infocom*, 1997.