Caching-Enhanced Scalable Reliable Multicast

Carolos Livadas*
BBN Technologies

Idit Keidar
Dept. of Electrical Engineering, Technion

Abstract

We present the Caching-Enhanced Scalable Reliable Multicast (CESRM) protocol. CESRM augments the Scalable Reliable Multicast (SRM) protocol [4,5] with a caching-based expedited recovery scheme. CESRM exploits the packet loss locality occurring in IP multicast transmissions in order to expeditiously recover from losses in the manner in which recent losses were recovered. Trace-driven simulations show that CESRM reduces the average recovery latency of SRM by roughly 50% and, moreover, drastically reduces the overhead in terms of recovery traffic and control messages.

1. Introduction

Developing scalable reliable multicast protocols is challenging, due to the requirements to scale to large multicast groups, to cater to dynamic memberships and changing networks, and to minimize the recovery overhead. A number of retransmission-based reliable multicast protocols [5, 7–9, 13, 14] have been designed to address these challenges, beginning with the seminal *Scalable Reliable Multicast (SRM)* protocol [4,5]. Such protocols use retransmissions in order to recover from losses. In SRM, packet recovery is carried out as follows. Upon detecting a loss (*e.g.*, by observing a sequence number gap in the stream of packets received from a given source), a receiver multicasts a retransmission request for the missing packet. In response, any member of the reliable multicast group that has the requested packet may retransmit it, using multicast. In order to minimize the number of requests and replies (*i.e.*, retransmissions) that are multicast per loss, SRM employs a suppression mechanism that relies on appropriately delaying the transmission of requests and replies. This mechanism causes loss recovery in SRM to be delayed by several network round-trip times.

* This work was conducted while Carolos Livadas was pursuing his Ph.D. at the Theory of Distributed Systems Group at the Lab. for Computer Science at MIT.

All the previously suggested retransmission-based protocols that we are familiar with (including SRM) treat each packet loss independently and run the recovery process anew for each loss. Our work is motivated by the observation that packet losses in IP multicast transmissions are not independent [1,6,15–17]. Thus, in the case of SRM, there is no need to repeat the suppression mechanism for each loss. Rather, the recovery of later losses can be expedited based on decisions made in the recovery of earlier ones.

We present the Caching-Enhanced Scalable Reliable Multicast (CESRM) protocol, which augments the functionality of SRM with a caching-based expedited recovery scheme. CESRM's expedited recovery scheme operates in parallel with SRM's recovery scheme. In this scheme, each receiver caches the requestor/replier pairs that carry out the recovery of its recent losses and uses this information to select an appropriate requestor/replier pair to carry out the expeditious recovery of each new loss. We henceforth refer to requestor and replier of the selected pair as the *expeditious requestor* and *expeditious replier*, respectively. Thus, upon detecting a loss, if a receiver considers itself to be the expeditious requestor, then it initiates an expedited recovery for the given packet by immediately unicasting an expedited request to the expeditious replier. Upon receiving this request, the expeditious replier immediately multicasts the requested packet. Since neither the expedited request nor the expedited reply is delayed, the packet is recovered much faster than with SRM's recovery scheme. In some cases, the expedited recovery may fail either due to packet loss or because the replier to which the expedited request is sent does not have the given packet. In such cases, CESRM falls back on SRM's usual recovery scheme.

Like SRM, the basic CESRM protocol is an end-to-end protocol that does not assume any intelligent network support beyond IP multicast. Therefore, as with SRM, every retransmission is multicast to the entire group (multicast tree), even if only a subset of the receivers lose the packet. Router-assisted protocols [8,12,13] eliminate this drawback by modifying the underlying IP multicast protocol to allow routers to forward requests to designated repliers, and also to *subcast* packets, *i.e.*, to send packets only to receivers that reside on certain subtrees of the IP multicast tree. Adopting this approach, we also present a router-assisted version of CESRM that exploits such router capabilities (if present) in order to achieve localized recovery. The router-assisted version of CESRM is more "light-weight" than other router-assisted protocols in that it requires less functionality in the underlying routers.

We use trace-driven simulations to evaluate CESRM's performance and compare it to that of SRM. In these simulations, we consider CESRM in its simplest form, where router-assistance is unavailable. Our results show that CESRM reduces the average recovery time of SRM by roughly 50%. Furthermore, CESRM sends fewer packet retransmissions: it sends between 30% and 80% the number of retransmissions sent by SRM. Finally, CESRM sends roughly as many control packets as SRM, but a large percentage of these are unicast whereas all of SRM's control packets are multicast. So all in all, CESRM's overhead is significantly smaller than that of SRM.

This paper is organized as follows. Section 2 describes SRM, Section 3 describes CESRM, and Section 4 evaluates CESRM's performance through trace-driven simulations. Finally, Section 5 concludes the paper.

2. Scalable Reliable Multicast

We now give a brief overview of the Scalable Reliable Multicast protocol of Floyd *et al*. [4, 5]. SRM is an application-layer protocol implemented atop the IP multicast best-effort communication primitive. SRM consists of two functional components: i) *session message exchange*, and ii) *packet loss recovery*.

The hosts that are members of the multicast group exchange *session messages* so as to estimate their distance to each other — inter-host distances are quantified by the one-way transmission latency from one host to another. Moreover, by include information about which packets have been received from each transmission source, session messages also assist in detecting packet loss. Since this aspect of SRM is not central to this paper, we skip its detailed description (for further details see [4, 5, 10, 11]).

SRM's *packet loss recovery scheme* is receiver-based. A receiver detects the loss of a packet in one of two ways: (1) by noticing a gap in the sequence numbers in the stream of packets it receives from a given source; or (2) by learning from a session message that another receiver has received the missing packet. Upon detecting the loss of a packet, the receiver engages in the loss recovery process. The recovery process is logically divided into asynchronous *rounds*. A round involves the transmission of a *repair request* (requesting the packet's retransmission) by a receiver that lost the packet, and the transmission of a *repair reply* (the retransmission of the packet) by either the source or another receiver that has received the packet. Since a recovery round may fail to recover the packet due to additional losses, several recovery rounds may be required in order to recover a given packet.

All requests and replies are sent using IP multicast. SRM uses a *suppression mechanism* in order to minimize the number of requests and replies transmitted during the recovery of a given loss. This mechanism is based on *delaying* the transmission of repair requests and replies, and *suppressing* their transmission if the same requests or replies are received from other hosts. The delay period is randomly chosen within a time interval that depends on that host's distance from the source of the lost packet (for requests) or from the requestor (for replies). We proceed by describing these mechanisms in more detail.

2.1. Scheduling repair requests

SRM uses two suppression techniques: *deterministic* and *probabilistic suppression*. Deterministic suppression dictates that the transmission time of a request be scheduled proportionately to the distance of the requestor to the source. Thus, hosts that are closer to the source have a better chance of suppressing their descendants in the underlying IP multicast tree. Probabilistic suppression dictates that the transmission time of a request be scheduled randomly within a particular time interval. Thus, hosts that are equidistant from the source probabilistically suppress each other. SRM's request scheduling parameters $C_1, C_2 \in \mathbb{R}^{\geq 0}$ control how aggressively deterministic and probabilistic suppression, respectively, are used.

Upon detecting the loss of a packet p, a host h schedules a request by setting a *request timeout timer* to a value uniformly chosen within the interval $[C_1\hat{d}_{hs}, (C_1{+}C_2)\hat{d}_{hs}]$, where $\hat{d}_{hs}$ is h's distance estimate to the source s of p. Upon the expiration of the request timeout timer for p, h multicasts a repair request for p, and also schedules a new request for p for the next recovery round. The new request timeout timer is now set to a uniformly chosen value in the interval $2^k[C_1\hat{d}_{hs}, (C_1 + C_2)\hat{d}_{hs}]$, where k is the number of times that a request for p has already been scheduled. If h receives a request for packet p while h has a scheduled request for p, then the scheduled request for p is rescheduled to the next recovery round, by resetting the request timeout timer to a uniformly chosen value in the interval $2^k[C_1\hat{d}_{hs}, (C_1 + C_2)\hat{d}_{hs}]$. Note that whenever the request is rescheduled, the interval is doubled (since k increases).

Requests should only be backed off once per recovery round, even when multiple requests are sent in the same recovery round. To this end, SRM designates a *back-off abstinence period*, which is a time interval during which the request's timeout is not backed off again. Once h reschedules a request for p, following either the transmission or the reception of another request for p, it also sets a *back-off abstinence* timeout to the value $2^k C_3 \hat{d}_{hs}$, where k is the back-off used to schedule the request, and $C_3 \in \mathbb{R}^{\geq 0}$ is a parameter. Requests for p received prior to the expiration of the back-off abstinence timeout for p are discarded; they are considered to pertain to the prior recovery round. Thus, the back-off abstinence period prevents the request from being backed-off multiple times during the same recovery round.

As an aside, we note that our designation of the abstinence period departs slightly from the original description of SRM [4, 5], which sets the back-off abstinence timeout to half the time to the next request.[1] We have replaced the half with a parameter, in order to allow more tuning freedom.

2.2. Scheduling repair replies

In scheduling replies, deterministic and probabilistic suppression operate in a similar fashion. First, replies are scheduled proportionately to the reply scheduling parameter $D_1 \in \mathbb{R}^{\geq 0}$ and the distance of the replier to the requestor. Secondly, replies are scheduled within reply intervals whose width is proportional to the reply scheduling parameter $D_2 \in \mathbb{R}^{\geq 0}$ and the distance between the replier and the requestor.

Let h be a host that has either sent or received the packet p and receives a repair request for p from a host h'. Upon receiving this repair request for p, h schedules the transmission of a repair reply for p, by setting a *reply timeout timer* to a uniformly chosen value in the interval $[D_1 \hat{d}_{hh'}, (D_1 + D_2)\hat{d}_{hh'}]$, where $\hat{d}_{hh'}$ is h's distance estimate to h'. Upon the expiration of this reply timeout for p, h multicasts a repair reply for p. If a reply for the packet p is received while a scheduled reply for p is awaiting transmission, then the scheduled reply for p is canceled.

Once h either receives or sends a repair reply for p, it observes a *reply abstinence period*. During this period, h considers a reply for p to be *pending* and, thus, refrains from scheduling additional replies for p; requests for p that are received during this period are simply discarded. The extent of the reply abstinence period for p is dictated by a *reply abstinence* timeout. Upon either receiving or sending a repair reply for p, h sets the reply abstinence timeout for p to the value $D_3 \hat{d}_{hh'}$, where $D_3 \in \mathbb{R}^{\geq 0}$ is SRM's reply abstinence parameter. Reply abstinence periods prevent duplicate requests pertaining to a given recovery round for p from generating duplicate replies.

The use of these suppression techniques introduces a performance trade-off. While choosing large values for the scheduling parameters C_1, C_2, D_1, and D_2 affords more effective suppression, it also prolongs the packet recovery and results in larger recovery latencies.

3. Caching-Enhanced Scalable Reliable Multicast (CESRM)

In addition to SRM's recovery scheme, CESRM implements a caching-based expedited recovery scheme. In this scheme, members of the reliable multicast group attempt to expeditiously recover losses based on how recent losses were recovered. Hosts cache the requestor/replier pairs involved in the recovery of recent losses from each source. Upon detecting a loss, an expedited requestor/replier pair for this loss is chosen according to the cached information pertaining to the lost message's source. The expeditious requestor unicasts a request to the expeditious replier, which in turn, multicasts the packet. Expedited recoveries are not delayed for the purpose of suppression. Thus, when successfully recovering a packet, they result in minimal recovery latency, and suppress the requests and replies scheduled by SRM's usual recovery scheme. However, expedited recoveries may fail either due to further packet losses or because the replier to which the expedited request is sent has shared the loss and is thus incapable of retransmitting the packet. In such cases, CESRM falls back on SRM's usual recovery scheme.

We now proceed to describe in detail how CESRM works. Section 3.1 explains how the cache is managed. In Section 3.2, we explain how the cached information is used for expedited recovery. Section 3.3 presents an improvement to CESRM that exploits intelligent router capabilities (if present). Finally, in Section 3.4, we provide a simple analysis of CESRM's expedited and non-expedited recovery delays.

3.1. Caching Requestor/Replier Pairs: Basic Approach

Each host maintains a collection of per-source requestor/replier caches, one for each source from which it receives packets. For simplicity of the exposition, we present the protocol for a single source IP multicast transmission, where each receiver h maintains a single cache for the source s. The cache contains the requestor/replier pairs that carried out the recovery of the most recent packets from s that were lost by h. More precisely, the cache consists of tuples of the form $\langle i, q, \hat{d}_{qs}, r, \hat{d}_{rq} \rangle$, where i is a packet sequence number, q is a requestor, $\hat{d}_{qs}$ is q's distance estimate to s, r is a replier, and $\hat{d}_{rq}$ is r's distance estimate to q.

When a packet is requested and/or retransmitted multiple times, multiple plausible requestor/replier pairs may arise.

[1] SRM [4, 5] also suggests an alternative approach for setting the abstinence period using message annotations, which we do not consider in this paper.

In such a case, h caches only the *optimal* requestor/replier pair for the given packet. We consider a requestor/replier pair to be optimal when it affords the minimum *recovery delay*; in our work, we define a packet's recovery delay to be the sum of the distance estimate from the requestor to the source and the round-trip distance from the requestor to the replier, *i.e.*, $\hat{d}_{qs} + 2\hat{d}_{rq}$. This definition gives preference to requestors that are closer to the source and to repliers that can provide the smallest recovery latency.

Optimal requestor/replier pairs are ascertained by simply annotating request and reply packets with the appropriate information. In particular, each request packet is annotated by the requestor and its distance to the source of the packet being requested, *e.g.*, $\langle q, \hat{d}_{qs} \rangle$. Each reply packet is annotated with the requestor that instigated the reply, this requestor's distance to the source, the replier, and this replier's distance to the requestor, *e.g.*, $\langle q, \hat{d}_{qs}, r, \hat{d}_{rq} \rangle$.

Host h updates the contents of its cache upon receiving replies. When h receives a reply for a packet i, then if h did not suffer the loss of packet i, the reply is discarded. The reply is also discarded if the cache is full and packet i is less recent than all the packets for which requestor/replier pairs are already cached. Otherwise, h processes the reply and updates the cache contents as follows: If no requestor/replier tuple pertaining to packet i is already cached, then the recovery tuple annotating the reply is cached; if the cache is full, it replaces the tuple pertaining to the least recent packet. If a requestor/replier tuple for packet i is already cached, then the cached tuple is updated to reflect the optimal requestor/replier pair.

3.2. Expedited Recoveries

Upon detecting the loss of packet i, a host h schedules a request for packet i using the usual SRM recovery mechanism. In addition, h consults the optimal requestor/replier cache for the source s in order to determine whether it should also act as the *expeditious requestor*.

The host h examines the optimal requestor/replier pairs it has cached and determines which such pair is the most appropriate to carry out an expedited recovery for packet i. Several policies may be used for selecting this expeditious requestor/replier pair. The *most recent loss* policy is to select the optimal requestor/replier pair that carried out the recovery of the most recent packet that h lost and has since recovered. The *most frequent loss* policy is the one in which the expeditious requestor/replier pair is chosen to be the pair that appears most frequently in the optimal requestor/replier pair cache. Other more sophisticated policies for selecting the expeditious requestor/replier pair may indeed be more effective than either of these policies.

Suppose that $\langle q, r \rangle$ is the *expeditious recovery pair* dictated by the expedition requestor/replier policy used by h. If h is the requestor of the expedited recovery pair $\langle q, r \rangle$, *i.e.*, $h = q$, then h schedules the transmission of an expedited request for the packet i for `REORDER-DELAY` time units in the future, where `REORDER-DELAY` is a CESRM parameter. This delay serves to prevent the transmission of extraneous expedited requests when packets are temporarily presumed missing due to packet reordering. If packet i is received prior to the transmission time of the expedited request, then h cancels its expedited request for packet i. Otherwise, h unicasts the expedited request to the expeditious replier r.

Upon receiving this expedited request for packet i, the expeditious replier r immediately multicasts an expedited reply for packet i, provided that it has previously either sent or received packet i and a reply for packet i is neither scheduled nor pending.

3.3. Router-Assisted Local Recovery

Up to this point, our presentation of CESRM has assumed no network support beyond a best-effort multicast service like IP multicast. While this approach makes the protocol more readily deployable, it has a disadvantage in terms of performance: the retransmission of every lost packet is multicast to the entire multicast group. This drawback is even more significant in SRM, where packet requests are also multicast to the entire group.

In order to remedy this shortcoming (of SRM), several router-assisted reliable multicast protocols have recently been proposed [8, 12, 13]. Such protocols assume that the underlying IP multicast routers have enhanced functionalities that allow them to forward packet requests to designated repliers and to *subcast* packet retransmissions to a subtree of the IP multicast tree. Such protocols, *e.g.*, the *Light-weight Multicast Services (LMS)* protocol [13], appoint designated hosts (called repliers) to reply to requests originating within particular subtrees of the underlying IP multicast tree. In the case of LMS, for example, each router in the multicast tree maintains a replier link onto which it forwards requests that originate within the subtree rooted at that router. Thus, every request originating in a certain subtree is forwarded by the router at the root of that subtree to that subtree's designated replier. Subsequently, the replies to such requests are unicast to the aforementioned routers, which in turn subcast the replies downstream.

In effect, router-assisted reliable multicast protocols use the enhanced IP multicast router functionality to introduce a recovery hierarchy. This hierarchy is very effective in achieving localized recovery and, thus, reducing recovery exposure. However, it may not fare well in highly dynamic environments where reliable multicast group members may either leave or crash unexpectedly. In such cases, the replier state maintained by the IP multicast routers becomes stale and must be updated. Such updates may prolong and even inhibit packet loss recovery.

CESRM's caching-based expedited recovery scheme, as presented above, effectively establishes a similar hierarchy of repliers. However, instead of pre-designating repliers and making them known to the routers, CESRM determines the appropriate repliers on-the-fly according to the cached information. Thus, CESRM's choice of repliers evolves to match changes in the group membership resulting from member joins, leaves, and crashes. Although this evolution may take time, packets continue to be recovered in the interim, because when expedited recoveries fail, losses are still recovered by SRM's recovery scheme. It is important to note that the expeditious requestor/replier selection policy affects how fast CESRM's expedited recovery scheme adapts to membership and topology changes.

Using minimal additional IP multicast router functionality, CESRM's expedited recovery scheme can also achieve localized recovery. In particular, routers need only be augmented to: i) annotate reply packets with their *turning point routers*, *i.e.*, the routers at which reply packets are received from and forwarded on downstream links with respect to the source of the original packet, and ii) subcast expedited reply packets downstream. This functionality is nearly identical to that of LMS [12, 13], with the exception that LMS requires routers to maintain replier state.

CESRM may exploit such extra router functionality as follows: Recovery tuples may be augmented to include the turning point router involved in the recoveries of the respective packets. By annotating each expedited request with the pertinent recovery tuple and the pertinent turning point router, the resulting expedited reply may be unicast to the particular turning point router, which may subsequently subcast the reply downstream. Since IP multicast routers need not maintain replier state, our scheme offers lighter-weight local recovery than LMS. Moreover, by employing SRM as a fall-back recovery scheme, CESRM remains robust in highly dynamic and faulty environments, whereas LMS does not.

3.4. Expedited vs. Non-Expedited Recoveries

We now compare the recovery latency of CESRM's expedited and non-expedited recovery schemes. In this section, we let $\overline{d}$ and $\overline{RTT} = 2\overline{d}$ be upper bounds on the one-way and round-trip distance (delay) between any two members of the reliable multicast group.

We first consider successful first-round non-expedited recoveries. Since requests and replies are scheduled uniformly within the request and reply intervals, a rough upper bound on the average latency of a successful first-round non-expedited recovery is given by:

$$(C_1 + 1/2C_2)\overline{d} + \overline{d} + (D_1 + 1/2D_2)\overline{d} + \overline{d}. \qquad (1)$$

This delay is exhibited by the scenario in which both the request and reply are scheduled for transmission at the midpoint of the request and reply scheduling intervals, respectively. This is a rough upper bound for two reasons. First, $\overline{d}$ is an upper bound on the inter-host transmission latencies, and some of the latencies may be smaller. Second, since multiple requests may be scheduled per loss, the request that instigates a packet's recovery is either sent or received with higher probability in the first half of the request interval. This is similarly true for replies.

In contrast, an upper bound on the recovery latency of a successful expedited recovery of CESRM is given by:

$$\texttt{REORDER-DELAY} + 2\overline{d} = \texttt{REORDER-DELAY} + \overline{RTT} \quad (2)$$

Given the typical SRM scheduling parameter values used by Floyd *et al.* [4, 5] of $C_1 = C_2 = 2$ and $D_1 = D_2 = 1$, the rough upper bound on the average recovery latency of a successful first-round non-expedited recovery of CESRM is $6.5\ \overline{d}$, or $3.25\ \overline{RTT}$. Assuming that the delay REORDER-DELAY is negligible compared to the latency, *i.e.*, REORDER-DELAY $\ll \overline{RTT}$, CESRM's recovery latency for packets recovered by expedited rather than first-round non-expedited recoveries is reduced by roughly $2.25\ \overline{RTT}$.

In the next section, we study the average recovery latencies afforded by both SRM and CESRM in simulations based on real IP multicast transmissions. We show that the average recovery time of first-round recoveries in SRM indeed varies between $1.5\ \overline{RTT}$ and $3.25\ \overline{RTT}$; this corresponds to the average recovery latency of CESRM's non-expedited first-round recoveries. Moreover, the average difference in latency between expedited and non-expedited first-round recoveries in CESRM varies between $1\ \overline{RTT}$ and $2.5\ \overline{RTT}$.

4. Evaluation Through Trace-Driven Simulations

We evaluate the performance of SRM and CESRM using trace-driven simulations in NS2 [3]. Our simulations reenact the 14 IP multicast traces of Yajnik *et al.* [15] and, thus, capture the packet loss locality exhibited in the actual IP multicast transmissions. We contrast the performance of CESRM against that of SRM. We consider CESRM in its simplest form, where router-assistance is unavailable.

We begin this section by describing the 14 IP multicast transmission traces of Yajnik *et al.* [15] and the manner in which we estimate the links on which each loss occurs. Next, we describe the simulation setup. Finally, we present our simulation results.

Table 1 IP Multicast traces of Yajnik *et al*. [15].

	Source & Date	# of Rcvrs	Tree Depth	Period *(msec)*	Duration *(hr:min:sec)*	# of Pkts	# of Losses
1	RFV960419	12	6	80	1:00:00	45001	24086
2	RFV960508	10	5	40	1:39:19	148970	55987
3	UCB960424	15	7	40	1:02:29	93734	33506
4	WRN950919	8	4	80	0:23:31	17637	10276
5	WRN951030	10	4	80	1:16:02	57030	15879
6	WRN951101	9	5	80	0:55:40	41751	18911
7	WRN951113	12	5	80	1:01:55	46443	29686
8	WRN951114	10	4	80	0:51:23	38539	11803
9	WRN951128	9	4	80	0:59:56	44956	33040
10	WRN951204	11	5	80	1:00:32	45404	16814
11	WRN951211	11	4	80	1:36:42	72519	44649
12	WRN951214	7	4	80	0:51:38	38724	20872
13	WRN951216	8	3	80	1:06:56	50202	37833
14	WRN951218	8	3	80	1:33:20	69994	43578

4.1. IP Multicast Traces

We use 14 IP multicast transmission traces of Yajnik *et al*. [15]. These traces involve single-source IP multicast transmissions in which packets are transmitted from the source at a constant rate. These packets are transmitted using IP multicast to a subset of 17 research community hosts spread out across the US and Europe.

The data collected from each of the IP multicast transmissions involves per-receiver sequences, each of which indicates which packets were received and the order in which they were received by the respective receiver. These per-receiver sequences do not include the packet reception times. Yajnik *et al*. also provide the IP multicast tree topology for each of the IP multicast transmissions. This topology is presumed to be static (fixed) throughout the duration of the IP multicast transmission. Table 1 lists the source, number of receivers, IP multicast tree depth, packet transmission period, transmission duration, the number of packets transmitted, and the number of losses suffered for each of the 14 traces. For more information regarding the traces, see [15].

Consider an IP multicast transmission trace. Let $k \in \mathbb{N}$ be the finite number of packets transmitted during the trace and R be the finite set of receivers of the IP multicast transmission. For $I = \{1, \dots, k\}$ and $i \in I$, we refer to the i-th packet transmitted during the IP multicast transmission as packet i. As is traditionally done in the literature [1, 6, 15, 16], we represent the trace data by per-receiver binary sequences of length k. We define a mapping *loss* $: R \to (I \to \{0, 1\})$, such that, for $i \in I$ and $r \in R$, *loss*$(r)(i) = 1$, if receiver r suffered the loss of packet i, and *loss*$(r)(i) = 0$, otherwise.

We represent the IP multicast tree, along which the k packets of the IP multicast transmission are disseminated, as a tuple $T = \langle N, s, L\rangle$ consisting of a set of nodes N, a root node $s \in N$, and a set of directed edges $L \subseteq N \times N$. The elements N, s, and L of T are further constrained to form a directed tree rooted at s in which all edges in L are directed away from s, there is a unique simple path from s to each other node in N, and the elements of R are exactly the leaf nodes of the tree (and, consequently, $R \subseteq N$). The root node s corresponds to the source of the IP multicast transmission, the internal nodes of T correspond to the IP multicast capable routers of the network that are used to disseminate the packets transmitted by s, and the leaf nodes of T correspond to the receivers of the IP multicast transmission. The edges of T correspond to the communication links that connect the source, routers, and receivers of the IP multicast transmission. We henceforth also refer to the edges of T as links.

4.2. Estimating the Links Responsible for the IP Multicast Transmission Losses

We estimate the links responsible for each loss suffered during the IP multicast transmission based on the IP multicast tree topology and the loss pattern observed in the IP multicast transmission trace for the respective packet. Each loss pattern observed in a trace may be the result of losses on either a single or a combination of IP multicast tree links. For example, the loss pattern involving all receivers may result from either a single loss on the link leaving the source, losses on each of the links leading to the receivers, or from a number of other combinations. We select a particular combination of links to represent each instance of a loss pattern based on the probability that a packet is dropped on exactly the links in each combination. We estimate this probability by first estimating the probability that a packet is dropped on each link of the IP multicast tree, *i.e.*, the link loss rates.

Letting $l_{nn'} \in L$ be the link that connects the nodes n and n', where n is the parent of n', we define $p(l_{nn'})$ to be the probability that a packet is dropped along $l_{nn'}$ given that the packet is received by n. The probabilities $p(l_{nn'})$, for $l_{nn'} \in L$, can be estimated either by the method of Yajnik *et al*. [15] or the maximum-likelihood estimator method of Cáceres *et al*. [2]. For the traces used in this paper, we found that both methods yield very similar link loss probability estimates. The simulations below are based on the estimates obtained using the former method.

Given the IP multicast tree, it is straightforward to deduce the set of link combinations that result in any loss pattern observed in the trace. We assume that the probability of a packet being dropped on a link is independent of it being dropped on any other link. We compute the probability of occurrence of a particular link combination as the product of the probabilities of a packet being dropped on the links in the combination and successfully forwarded on the links leading to the links in the combination.

More precisely, consider an observed loss pattern x. Let C_x be the set of all possible link combinations resulting in x, L_c be the set of links in a combination $c \in C_x$, and U_c be the set of links that are neither in L_c nor downstream of any of the links in L_c. Presuming that the probabilities of loss along the different links of the IP multicast tree are mutually independent, the probability of occurrence of the link combination c is estimated by $p(c) = \prod_{l \in L_c} p(l) \cdot \prod_{l' \in U_c} (1 - p(l'))$. Thus, the probability that the observed loss pattern x results from the link combination c as opposed to the other combinations in C_x is given by $p_{C_x}(c) = p(c) / \sum_{c' \in C_x} p(c')$.

We select the link loss combination to represent an instance of the loss pattern x in the trace based on the probabilities of occurrence of all link loss combinations resulting in x. For 13 of the 14 traces we consider, more than 90% of the link combinations selected to represent the losses occur with probabilities exceeding 95%, often very close to 100%. For the remaining trace, 85% of the link combinations selected to represent the losses occur with probabilities that exceed 98%. Thus, our estimates of the links responsible for the losses suffered in each trace are predominantly accurate.

Based on the link loss combinations selected to represent each loss suffered in the trace, we define the *link trace representation* to be the mapping *link* : $R \to (I \to L \cup \perp)$, such that, for $r \in R$ and $i \in I$, *link*$(r)(i)$ is an estimate of the link responsible for the loss of packet i by receiver r, if receiver r suffered the loss of packet i, and *link*$(r)(i) = \perp$, if receiver r did not suffer the loss of packet i.

4.3. Simulation Setup

In our simulations, we use the most recent loss expedited requestor/replier selection policy. According to this policy, the expedited requestor/replier pair is the optimal requestor/replier pair of the most recent loss that has already been recovered. In [10], we analyzed the traces of Yajnik *et al*. [15] and found that the most recent loss policy outperforms the most frequent loss policy. This is because, more often than not, the location of a loss is correlated to a higher degree with the location of the most recent loss than with the locations of less recent losses. An additional advantage of the most recent loss policy is the simplicity of its implementation; receivers need only cache a single optimal requestor/replier pair.

For a given trace, our simulation involves setting up the IP multicast tree T and disseminating k packets from the root of the tree to the tree's leaf nodes. Recall that the IP multicast tree is presumed to remain fixed throughout the duration of the IP multicast transmission.

Since the IP multicast trace information of Yajnik *et al*. [15] contains no link delay or bandwidth information, we had to synthetically choose values for these parameters. We chose the bandwidth of each link in T to be 1.5 *Mbps*. We assume that payload carrying packets, *i.e.*, original packets and retransmissions, are 1 *KB* in size, and control packets (*i.e.*, packet retransmission requests and session messages) are 0 *KB*. Since the IP multicast transmission period of any of the IP multicast transmission traces of Yajnik *et al*. [15] is either 40 *ms* or 80 *ms*, the bandwidth required for the IP multicast transmissions is either 200 *Kbps* or 400 *Kbps*. Thus, our choice of 1.5 *Mbps* for the link bandwidth is sufficient to carry the IP multicast transmission data.

We ran our simulations with three different link delays: 10 *ms*, 20 *ms*, and 30 *ms*, where in each simulation all the links had the same delay. The results with the three different choices were very similar. We therefore include here only the results obtained with a link delay of 20 *ms*. Since the depth of the IP multicast tree involved in each of the IP multicast traces of Yajnik *et al*. [15] ranges from 3 to 7, the RTTs between the source and receivers in each trace ranges from 60 *ms* to 420 *ms*. The range of these inter-host RTT values is reasonable for hosts spread out across the US and Europe.

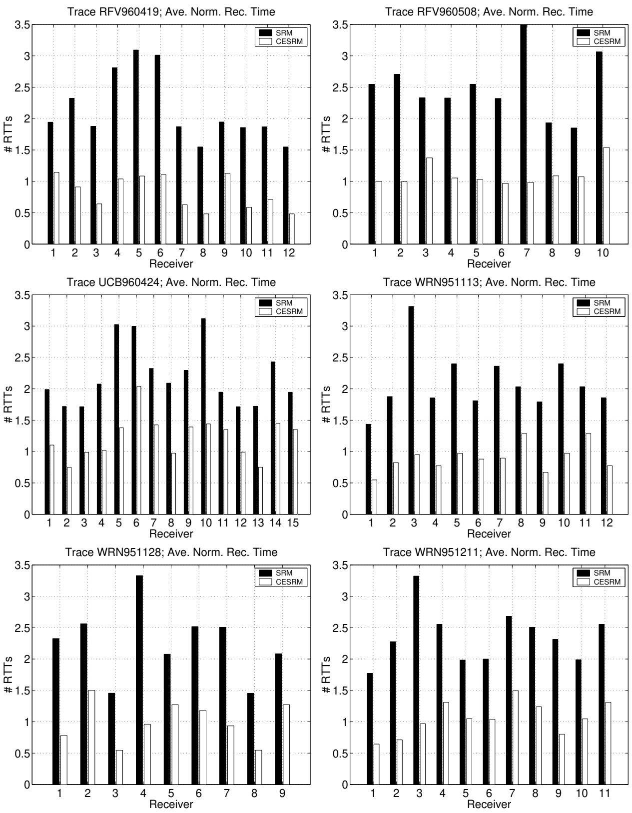
The simulation of SRM is carried out with the scheduling parameter settings $C_1, C_2 = 2$, $C_3 = 1.5$, $D_1, D_2 = 1$, and $D_3 = 1.5$. These correspond to the typical SRM parameter settings used by Floyd *et al*. [4,5]. Since packets are not reordered in our simulations, we use a `REORDER-DELAY` of 0 *sec*.

Session packets are transmitted with a period of 1 *sec*. In order to focus our attention on the performance of CESRM packet loss recovery scheme, rather than that of the inter-host distance estimation scheme through session packet exchange, we presume that the session packet exchange is lossless. Since none of the session packets are dropped throughout our simulation, the inter-host distances are accurately and promptly calculated. Moreover, the IP multicast transmission is delayed sufficiently so that, prior to its beginning, receivers have a chance to exchange session messages and, thus, estimate their distances to each other.

We inject losses into the simulated IP multicast transmission according to the link trace representation *link*, which identifies estimates of the links responsible for the losses suffered by each receiver during the actual IP multicast transmission. By injecting losses in this fashion, we reproduce the packet loss pattern present in the actual IP multicast transmission.

In our simulations, we assume that packet loss recovery is lossless; that is, none of the recovery packets (control packets and retransmissions) are dropped. We chose to simulate lossless recovery because when message loss is considered, there is a larger variability in the results. In [10], we also simulated the protocols with control packets and retransmissions being dropped based on the link loss probability estimates computed in Section 4.2. As expected, the recovery latencies of both SRM and CESRM were slightly larger, and CESRM exhibited similar performance improve-

Figure 1 Per-receiver average normalized recovery times.



ments over SRM to those presented herein. Due to space limitations, we do not include these results here.

4.4. Simulation Results

Figures 1–4 present per-receiver results obtained with 6 typical traces of the 14 studied traces. The results obtained with the remaining traces were very similar. Due to space limitations, we depict only the results obtained for these 6.

Figure 1 presents the per-receiver average normalized recovery times achieved by SRM and CESRM. The recovery time of each receiver is normalized by that receiver's RTT estimate to the source, and is therefore given in units of RTT. From Figure 1, we can see that the caching-based expedited recovery scheme employed by CESRM substantially reduces the average normalized recovery time. For most of the receivers, CESRM's average recovery times are 40% to 70% (50% on average) smaller than SRM's.

Figure 2 depicts the difference in the average normalized recovery times between expedited and non-expedited recoveries of CESRM. Equations (1) and (2) presented in Section 3.4 predicted that, for the scheduling parameters used in our simulations, the difference between the average recovery times of expedited and non-expedited recoveries would be roughly bounded by $2.25\ \overline{RTT}$. Figure 2 indeed reveals that, in our simulations, the difference in the average normalized recovery latency between expedited and non-expedited successful recoveries ranges from $1\ \overline{RTT}$ to $2.5\ \overline{RTT}$.

Figure 2 Difference in average normalized recovery times between expedited and non-expedited recoveries of CESRM.

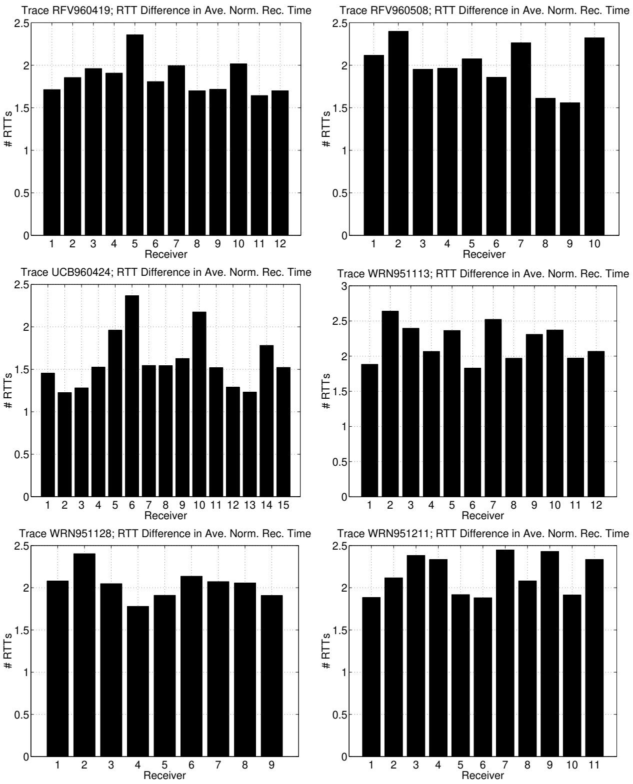


Figure 3 depicts the number of request packets sent by each of the receivers with SRM and CESRM. The bars corresponding to CESRM are split into two components. The white component corresponds to requests unicast by the expeditious requestor as part of CESRM's expedited recovery process; the gray component corresponds to requests that are multicast when CESRM falls back on SRM's recovery scheme. The source of the IP multicast transmission corresponds to receiver 0.

Figure 3 reveals that, for most receivers in each of the simulations, the number of requests sent by CESRM is less than the number sent by SRM. For some of the receivers the number of requests sent by CESRM exceeds that sent by SRM. Notably, however, a large portion of the requests sent by CESRM are unicast from particular requestors to particular repliers, rather than multicast to the entire group. Since unicast transmissions are substantially less costly than multicast transmissions, the overhead associated with sending requests in CESRM is substantially smaller than that incurred in SRM.

Figure 4 depicts the number of reply packets sent by each of the receivers with SRM and CESRM. The bars

Figure 3 Number of request packets for SRM and CESRM.

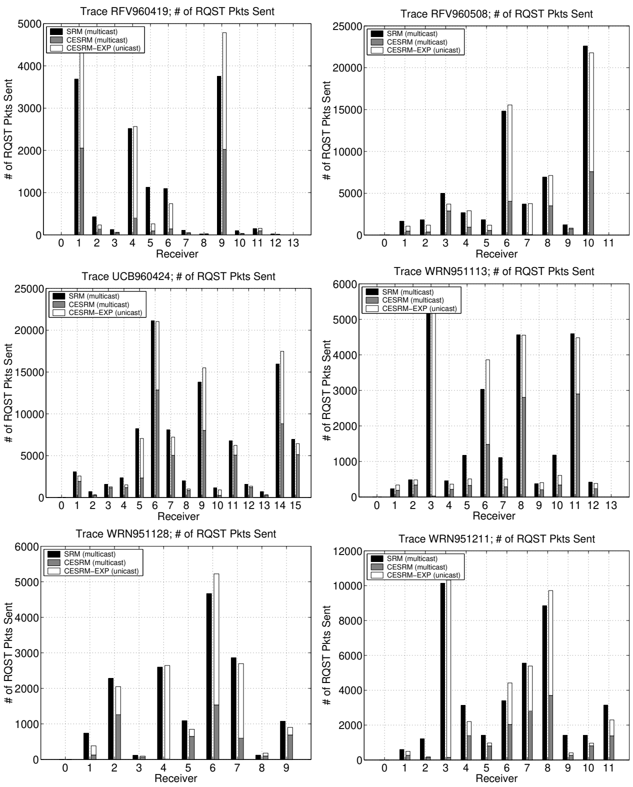
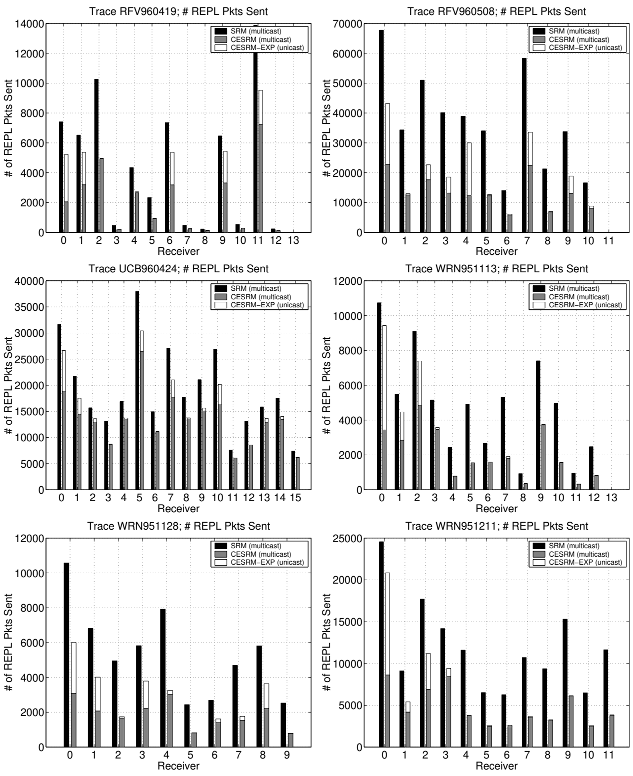


Figure 4 Number of reply packets for SRM and CESRM.



corresponding to CESRM are again split into two components. The white component corresponds to expedited replies multicast in response to expedited requests as part of CESRM's expedited recovery process; the gray component corresponds to replies that are multicast in response to non-expedited requests when CESRM falls back on SRM's recovery scheme. Again, the source of the IP multicast transmission corresponds to receiver 0.

As Figure 4 shows, for most receivers in each of the simulations, CESRM sends substantially fewer packet retransmissions (replies) than SRM. This is to be expected since successful expedited recoveries usually involve a single expedited reply, whereas SRM's suppression scheme may often result in duplicate replies per recovery. The reduction in the number of packet retransmission offers a significant improvement over SRM, since packet retransmissions carry payload, and are therefore generally larger than control messages.

The first plot in Figure 5 depicts the percentage of successful expedited recoveries achieved by CESRM for each of the traces. An expedited recovery is successful when the expedited request induces the transmission of an expedited reply. The percentage of successful expedited recoveries is given by the ratio of the number of expedited requests to the number of expedited replies transmitted during the simulation. We observe that more than 70% of the expedited recoveries are successful for all traces and as many as 80% are successful in all but two of the traces.

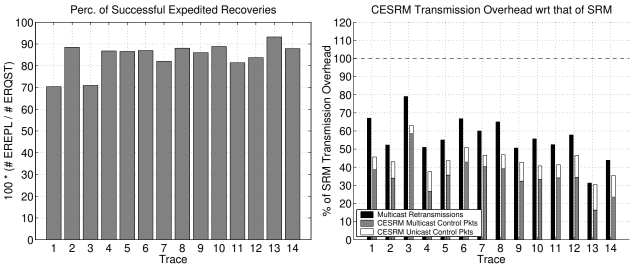
The second plot of Figure 5 depicts the transmission overhead of CESRM, as a percentage of the transmission overhead of SRM for each of the traces. This number amounts to 100% in traces where SRM and CESRM incur the same transmission overhead. We divide the transmission overhead into two parts; that incurred by retransmissions and that incurred by control packets. We also distinguish between unicast and multicast control packets. The transmission overhead of either CESRM or SRM is calculated by assigning a cost of 1 unit for transmitting a packet across any single link of the IP multicast tree and measuring the total number of such transmissions for each trace.

Figure 5 reveals that the transmission overhead incurred by CESRM retransmissions is substantially smaller than that incurred by SRM retransmissions. For all the traces, the retransmission overhead of CESRM is less than 80% of that of SRM. In 10 of the 14 traces, it is even less than 60% of that of SRM. In the case of control packets, CESRM's overhead is less than 52% of that of SRM for all but one of the traces. This demonstrates our claim that CESRM greatly reduces SRM's overall recovery overhead.

5. Conclusions

In this paper, we have presented CESRM, a scalable reliable multicast protocol, which augments SRM with a

Figure 5 CESRM performance.



caching-based expedited recovery scheme. CESRM exploits packet loss locality in order to reduce the overhead and improve the recovery time of SRM. Trace-driven simulations revealed that, indeed, CESRM reduces the average recovery time of SRM by an average of roughly 50%. We further observed that these performance gains do not introduce additional packet overhead. On the contrary, in all of our simulations, CESRM reduced the total number of packet retransmissions sent. Moreover, the overhead associated with sending control packets in CESRM was significantly smaller than that of SRM.

CESRM's expedited requests and replies resemble those occurring in router-assisted protocols [8, 12, 13], where requests are intelligently forwarded to designated repliers. Unlike these protocols, however, CESRM can be deployed over IP multicast without any special router support. We have also presented a router-assisted version of CESRM that makes use of intelligent router capabilities (if present) in order to limit the exposure of packet retransmissions. This protocol is more "light-weight" than router-assisted protocols like LMS [13], since it assumes fewer router capabilities. Moreover, unlike LMS, CESRM can continue to recover packets even while the multicast group is reconfiguring and the previously chosen repliers leave the group. This results from the fact that CESRM's recovery mechanism falls back on that of SRM.

Acknowledgments

We are grateful to Nancy Lynch for many helpful discussions. We thank Yajnik *et al*. [15] for making their multicast transmission traces available online. Finally, we thank the reviewers for their insightful comments and suggestions.

References

[1] J.-C. Bolot, H. Crépin, and A. Vega Garcia. Analysis of Audio Packet Loss in the Internet. In *Proc. NOSSDAV*, volume 1018 of *Lecture Notes in Computer Science*, pages 154–165, Apr. 1995.

[2] R. Cáceres, N. G. Duffield, J. Horowitz, and D. F. Towsley. Multicast-Based Inference of Network-Internal Loss Characteristics. *IEEE Transactions on Information Theory*, 45(7):2462–2480, Nov. 1999.

[3] K. Fall and K. Varadhan, editors. *The* ns *Manual (Formerly* ns *Notes and Documentation)*. The VINT Project, A Collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC, Jan. 2001. *ns* © is LBNL's Network Simulator, by S. McCanne and S. Floyd.

[4] S. Floyd, V. Jacobson, S. McCanne, C.-G. Liu, and L. Zhang. A Reliable Multicast Framework For Light-Weight Sessions And Application Level Framing. In *Proc. ACM/SIGCOMM*, pages 342–356, Aug. 1995.

[5] S. Floyd, V. Jacobson, S. McCanne, C.-G. Liu, and L. Zhang. A Reliable Multicast Framework For Light-Weight Sessions And Application Level Framing. *IEEE/ACM Transactions on Networking*, 5(6):784–803, Dec. 1997.

[6] M. Handley. An Examination of MBone Performance. Research Report RR-97-450, University of Southern California (USC)/Information Sciences Institute (ISI), Jan. 1997.

[7] H. W. Holbrook, S. K. Singhal, and D. R. Cheriton. Log-Based Receiver-Reliable Multicast For Distributed Interactive Simulation. In *Proc. ACM/SIGCOMM*, pages 328–341, Aug. 1995.

[8] D. Li and D. R. Cheriton. OTERS (On-Tree Efficient Recovery using Subcasting): A Reliable Multicast Protocol. In *Proc. IEEE/ICNP*, pages 237–245, 1998.

[9] J. C. Lin and S. Paul. RMTP: Reliable Multicast Transport Protocol. In *Proc. IEEE/INFOCOM*, volume 3, pages 1414–1424, Mar. 1996.

[10] C. Livadas. *Formal Modeling, Analysis, and Design of Network Protocols — A Case Study in Reliable Multicast*. Ph.D. Thesis, Dept. of Electrical Engineering and Computer Science, MIT, July 2003.

[11] C. Livadas and N. A. Lynch. A Formal Venture into Reliable Multicast Territory. In D. A. Peled and M. Y. Vardi, editors, *Proc. FORTE*, volume 2529 of *Lecture Notes in Computer Science*, pages 146–161, Nov. 2002.

[12] C. Papadopoulos. *Error Control for Continuous Media and Large Scale Multicast Applications*. Ph.D. Thesis, Washington University in St. Louis, 1999.

[13] C. Papadopoulos, G. Parulkar, and G. Varghese. An Error Control Scheme For Large-Scale Multicast Applications. In *Proc. IEEE/INFOCOM*, volume 3, pages 1188–1196, Mar. 1998.

[14] S. Paul, K. K. Sabnani, J. C. Lin, and S. Bhattacharyya. Reliable Multicast Transport Protocol (RMTP). *IEEE Journal on Selected Areas in Communications*, 15(3):407–421, Apr. 1997.

[15] M. Yajnik, J. Kurose, and D. Towsley. Packet Loss Correlation in the MBone Multicast Network. In *Proc. IEEE/GLOBECOM*, pages 94–99, Nov. 1996.

[16] M. Yajnik, S. B. Moon, J. Kurose, and D. Towsley. Measurement and Modeling of the Temporal Dependence in Packet Loss. In *Proc. IEEE/INFOCOM*, volume 1, pages 345–352, Mar. 1999.

[17] S. Yoon, K. Lee, E. Jin, J. Seo, J. Kim, and S. Choe. Reliable Multicast Considering the Temporal Dependence in Packet Loss. In *Proc. INET*, June 2001.