OTERS (On-Tree Efficient Recovery using Subcasting): A Reliable Multicast Protocol

Dan Li David R. Cheriton
Department of Computer Science
Stanford University
{dli, cheriton}@cs.stanford.edu

Abstract

This paper presents a reliable multicast protocol (OTERS) that organizes receivers into a fusion tree that matches the multicast delivery tree of the source and uses this tree to fuse NAKs and subcast retransmissions. The protocol requires two extensions to the current Internet network layer, namely multicast route backtracing and subcasting. We argue that these extension are simple, natural and low overhead extensions for the Internet. Simulations of six reliable multicast protocols demonstrate that our approach substantially reduces the recovery latency and the traffic load compared to previously proposed approaches.

1. Introduction

Designing a reliable multicast protocol that scales to audience sizes allowed by the Internet is a challenge. As the number of receivers grows to thousands or even millions, there is a significant probability of packet loss at one or more receivers on almost each transmission. Remulticasting lost packets over the entire multicast delivery tree can easily double the bandwidth load on the network. Moreover, with most packet loss due to congestion, particularly on tail circuits, multicast retransmission can lead to the self-defeating effect of increasing the packet loss. Other techniques such a cascaded unicast retransmission and deferred retransmission increase the time to recover from the packet loss, *recovery latency*, limiting the application domain. In general, the choices of techniques to date force a trade-off between high network traffic and high latency for recovery. Moreover, some approaches rely on complex extensions to routers entailing significant state, raising other concerns with scalability and practicality.

In this paper, we present a reliable multicast protocol called *OTERS* for *On-Tree Efficient Recovery using Subcasting*. This protocol requires only simple, low-overhead extensions to the network layer for multicast route backtracing and subcasting. Simulations of six reliable multicast protocols on NS [17] show that OTERS achieves a low recovery latency while imposing a low network traffic load on the network.

The rest of the paper is organized as follows. Section 2 describes the required network layer extensions. Section 3 describes the OTERS design and an extension to it called SCREAM. Section 4 presents the simulation results of six reliable multicast protocols. Section 5 discusses related work. Section 6 concludes this paper.

2. Network Extensions for OTERS

OTERS relies on two extensions to the network layer, multicast route backtracing and subcasting, for its performance. These extensions can be provided as simple modifications to IGMP traceroute [4] and IP encapsulation [14], respectively, as described below.

2.1. Multicast Route Backtracing

Multicast route backtracing is a facility by which a subscriber M to a multicast address G can determine the sequence of routers between a specified source S and itself. In particular, a multicast route backtracing query sent by M specifies G, S, and a hop limit K. The response contains a list of IP addresses of at most the last K hops of the multicast route leading to M, each corresponding to the incoming interface at which multicast packets from S to G should arrive. The hop limit allows incremental query up the multicast delivery tree and prevents implosion at the source.

Multicast route backtracing can be implemented as a subset of the functionality provided by IGMP traceroute (mtrace) [4] that allows any host H to trace the multicast route of host group G from a source S to a given destination M, and have the query response returned to an unicast or multicast address R. The mtrace response contains comprehensive per-hop information, e.g., interface addresses, timestamps and packet counts, etc. Note that mtrace runs on both shared-tree and source-specific multicast routing domains as well as a mix of the two.

Mtrace is designed for comprehensive network management and diagnostic programs. From the trace, one can infer not only the multicast routes but also detailed information such as what sources are sending to what groups at what rates. Security and privacy concerns may prevent this facility from being fully supported across network administration boundaries. For example, firewalls can block mtrace traffic at the border of their domains.

To address the security concerns, we propose a restricted form of mtrace, namely: (1) three IP addresses in a mtrace query, H, M and R, must be the same. (2) routers may leave most fields of the response template empty but fill the incoming interface field. (3) stub networks should not block outbound mtrace traffic[1]; transit networks should not block mtrace traffic. (4) a router may forward mtrace packets without appending its own response record. (5) to account for discontinuous records, each responding router copies the current TTL of the mtrace packet into its record.

The two mtrace semantics, comprehensive and restricted, can coexist. The network administrators can select the semantics for each router and requester, based on the local security policy. The use of mtrace for network management of multicast justifies its provision over just supporting OTERS.

2.2. Subcasting

Subcasting is a facility to multicast a packet over a subtree of the multicast delivery tree for a particular group, specified by the multicast address G and the router R at the root of this subtree. We refer to this as subcasting from R.

We propose a modified version of IP encapsulation [14] for implementing subcasting, allowing trust in the originality of the packet. In particular, denote an IP packet as (source, destination)[payload]. Then, a subcast packet appears as

(M, R) [(S, G) [payload]]

where group member M originates the packet, unicasts it to subroot router R. R swaps the inner IP header and the outer IP header instead of stripping the outer header[2] as in conventional encapsulation. The decapsulated packet is

(S, G) [(M, R) [payload]] .

[1] Note that if the source is in the stub network that blocks incoming mtrace traffic, routes close to the source (inside the stub network) may not be traced. Several highest-level subgroups may be formed outside the stub network and seek retransmissions directly from the source. The number of such subgroups is at most the fanout of the border gateway.

[2] If the inner header is a minimal forwarding header, only relevant fields are swapped.

R then multicasts the packet to all the outgoing interfaces that are used to deliver data from source S to multicast group G.

With a single-source model of IP multicasting as proposed in EXPRESS [7], subcasting is even simpler because every multicast and subcast packet must originate from the source. (See also Section 3.3.) Then subcasting can be implemented using loose source routing.

In both cases, the receiver can be assured of the source of the subcast packet up to its trust in the network infrastructure. In contrast, using conventional IP-in-IP encapsulation, the originator of a subcast packet is not indicated by the source address of the inner IP header and thus cannot be determined by the receiver. Because of this potential for source spoofing, multicast routers may refuse to forward these packets, thereby limiting its availability.

This mechanism requires allocation of a new IP protocol number and support in routers and end stations. However, the implementation is straight-forward, does not introduce new state in routers, is simpler than other proposals to extend the network layer for reliable multicast (Section 5), and is a key aspect in the efficiency of OTERS.

The subcasting supported in this form is useful for other multicast applications and management facilities, such as mobile IP, PIM-SM [3] and BGMP [16], as security concerns are taken further into account in the Internet.

Overall, these two extensions provide powerful yet simple, low-overhead facilities that are natural extensions to the Internet layer. Note that both extensions are slight modifications of existing or experimental facilities in the network layer where the modifications are required because of the growing security concerns in the Internet.

3. The OTERS Protocol Design

OTERS consists of the *Loss Recovery Protocol (LRP)* and the *Fusion Tree Formation Protocol (FTFP)*.

The FTFP organizes the multicast group members into the *OTERS fusion tree (OFT)*[3]. In this context, the term *subroot* refers to the root of an OFT subtree (a router). *Subgroup* refers to the group members in a subtree. Then, an OFT is an application-layer tree rooted at the multicast source in which each subtree is a subtree of the multicast routing tree and for each subtree there is a *designated receiver (DR)* that is a group member, located in this subtree, close to its subroot and knows the IP address of the subroot. Subgroups are hierarchical in that a subgroup member can be the DR of a descendant subgroup. As a special case, group members in the top-level subgroup use the source as the DR. A member can be a DR for multiple subroots. A

[3] The term *fusion tree* is used because this tree also serves to aggregate positive and negative acknowledgments.

DR's *top subroot* is the subroot nearest the source for which it is a DR.

We first describe the LRP, assuming an OTERS fusion tree has been established and then describe FTFP. The protocols are discussed in the one-to-many scenario. However, OTERS is equally applicable to many-to-many communication by applying the FTFP and LRP to each source.

3.1. The Loss Recovery Protocol (LRP)

The LRP handles retransmission of a data packet when a missing packet is detected, either by a gap in the sequence numbers or by timeout. The receiver missing a packet, referred to hereafter as *the requester*, immediately unicasts a NAK to its DR. If the requester is a DR, it requests retransmission be subcast from its top subroot. Otherwise, it request unicast retransmission to itself.

On receiving the NAK, the DR:

- requests the packet from its own DR if it did not itself receive the packet and has not yet requested the packet. (This case is unlikely because the DR is closer to the source than its subgroup members, and should normally detect the loss earlier.) Or,
- ignores the NAK if it has detected the same loss and is requesting subcast retransmission, or if a subcast repair for this packet has been sent or received recently[4]. Ignoring the NAK in this case avoids unnecessary repairs when the NAK is sent at the same time or shortly before the requester should receive the repair.
- otherwise, retransmits the segment.

For instance, LRP operation is illustrated in Figure 1.

NAKs are retransmitted with exponential backoff and an initial timeout as the round trip time to the DR[5]. When the elapsed time exceeds a threshold proportional to the one-way delay from the source, FTFP is re-invoked to locate a DR, thereby handling misbehaving DRs, and network and membership dynamics.

LRP efficiently repairs correlated losses using the OFT and subcast retransmission. When a packet drop occurs, resulting in loss at all receivers in the subtree underneath the point of drop, only the DR whose subroot is just above the drop point retransmits the lost packet. This subcast retransmission then repairs the loss in the entire subtree, assuming the retransmitted packet is not dropped on any links. The inefficient case where losses are independent yet a subcast repair is triggered is expected to be uncommon because

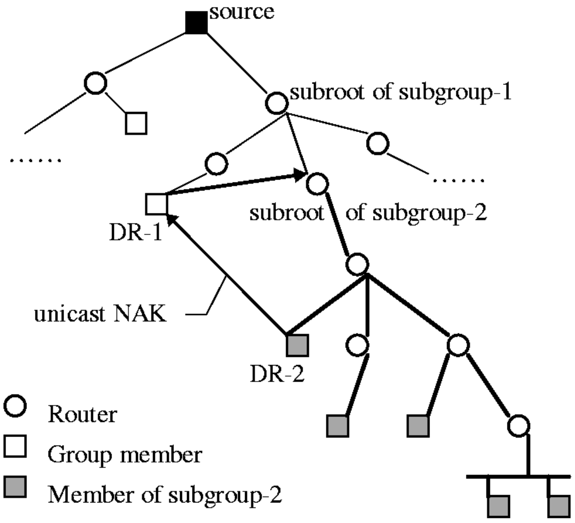

Figure 1. LRP. DR-2 unicasts a NAK to its own designated receiver, DR-1, after detecting a packet loss. DR-1 responds with a repair subcast from subgroup-2's top subroot, assuming DR-1 received this packet. Bold links indicate the path of the subcast retransmission.

studies, such as that of the MBone by M. Handley [6], show strong loss correlation. Thus, we argue that subcast transmissions do not typically retransmit to members that have received the packet. However, one should note that a DR high in the delivery tree with an unreliable link between it and the source could cause poor behavior.

LRP also makes efficient use of network resources by using the same routers (and thus same multicast forwarding state) and links for subcast retransmission as for original data transmission.

LRP is low latency because receivers closer to the source generally detect a loss earlier, and one of these receivers typically initiates the recovery process for an entire subtree. The NAK sent by one upstream receiver starts the recovery process for all the peers in the subtree, sometimes before the downstream receivers even detect the loss, further reducing the recovery time. Because of this *early-start* effect, the average loss recovery latency for an individual receiver may be smaller than the average round trip time to the retransmitter. (See also Section 4.4.)

3.2. FTFP: the Fusion Tree Formation Protocol

FTFP builds an OTERS fusion tree by: 1) incrementally identifying subroots on the multicast routing tree using multicast route backtracing, and 2) selecting a DR for each subroot. Each group member is assumed to know the IP address

[4]Here, "recently" means within the round trip time to the requester.

[5]For subcast repairs, the round trip time is the sum of a one-way unicast delay to the DR and a one-way subcast delay from the DR. For unicast repairs, it's the unicast round trip time to the DR.

of the multicast group G, multicast source S, the hop count from the source to itself H, the recent loss rate L, and the recent minimal transmission delay from the source to itself D[6].

An *FTFP packet* subcast by member M_i from a subroot R_{k_i} specifies four elements: (1) K_i – the hop distance from R_{k_i} to M_i, (2) D_i – the transmission delay from S to M_i, (3) L_i – the loss rate at M_i, and (4) the initial multicast TTL value of this packet. A receiver of this packet can infer its own hop distance from R_{k_i} by subtracting the remaining TTL from the initial TTL.

A member node i is defined as *superior* as a DR for subroot R_k compared to node j if the tuple $(L_i, D_i, K_{i,R_k}, M_i)$ corresponding to member i is less than (in dictionary order) the tuple $(L_j, D_j, K_{j,R_K}, M_j)$ from member j. where K_{i,R_k} is the hop distance from R_k to M_i, similarly for M_j. To prevent looping in the protocol, L_i and L_j (D_i and D_j) are considered equal if their difference lies within 20% of their mean. By selecting a minimal-loss-rate DR, we reduce the amount of unwanted retransmissions being subcast to the entire subtree.

A member node M performs the search for its DR as follows:

1. Unicast a mtrace query toward S with a hop limit of K to obtain the IP address of a potential subroot R_k that is K hops away from M. (K is initially set to 1.)

2. Subcast an FTFP packet from R_k to the group G. The FTFP packet should be delivered to all the group members in the subtree rooted at R_k, including M[7]. On receiving this subcast FTFP packet at M, set a timer equal to 2 times of the delay.

3. If an FTFP packet is received that specifies a node that is a superior DR to M for R_k before the above timer expires, record that node as its DR and stop the search. Otherwise, record itself as the DR for R_k, increment K and go to step 1.

A DR subcasts FTFP packets from its *top subroot* at some heartbeat rate. If a member does not receive an FTFP packet from its DR for 3 consecutive heartbeat periods, it selects the next best DR whose heartbeat has been arriving or resumes the search to actively find one.

On receiving an FTFP packet for a subroot R, M:

- ignores the packet if it has a DR and the DR's subroot is further downstream from R because this DR will respond as appropriate. Or,

- subcasts M's own FTFP packet from R if M is a superior DR to the sender for subroot R.

- otherwise, records the sender as its DR and stops any search process. (As a possible extension, M can also measure the round-trip time to its current DR and the candidate, and pick the closer one.)

Figure 2 shows one round of the search process.

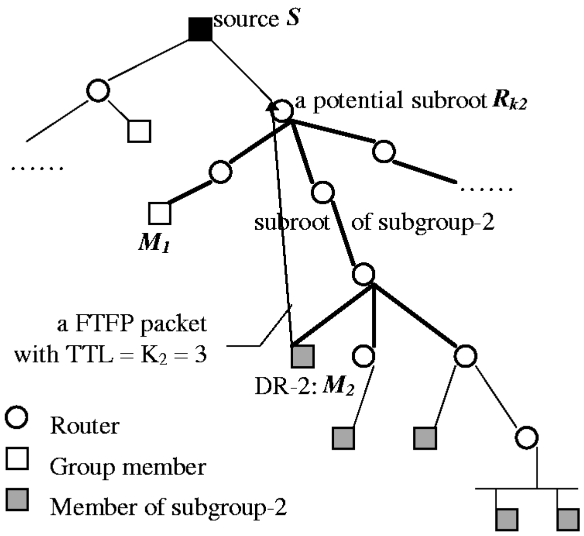

Figure 2. FTFP. M_2 is participating in the formation of a higher-level subgroup after being selected DR for subgroup-2. It subcasts a FTFP packet from potential subroot R_{k2} with a TTL of 3. Bold links experience the subcast transmission, with the TTL scoping preventing it from reaching the lower part of the tree. M_1 can be expected to respond with its own FTFP packet, so M_2 concludes that M_1 is a superior DR for R_{k2}, assuming the links to M_1 do not indicate higher delay or loss rate.

The protocol as specified assumes that all routers support multicast route backtracing and subcasting. To allow for routers that do not, i.e. *non-conforming* routers, M can increment K after failing to receive a response to its mtrace query until it does reach a responding router, thereby skipping over these non-conforming intermediate routers. The presence of non-conforming routers simply increases the size of some subgroups and reduces the granularity of subcast retransmission, making it less efficient. In the extreme,

[6] Recent minimal transmission delay is a more reliable measure of the topological distance than the average delay. H and D can be obtained from packets received from the source that contained the initial TTL and the timestamp. The source and receivers' clocks should be coarsely synchronized, e.g. within ten milliseconds.

[7] As an optimization, the TTL of the subcast packet can restrict its delivery so the links lower than M do not receive the subcast. However, the effectiveness of this optimization may be limited by administrative boundaries in some cases.

with no router support, OTERS degenerates to a single subtree equal to the multicast delivery tree and multicast retransmission from the source. Thus, incremental deployment of these router features is feasible. Increasingly wide-scale use of OTERS would simply increase the pressure for networks to upgrade routers to these capabilities.

An overloaded DR can force potential subgroup members to join the subgroup of another DR by sending heartbeats with a limited TTL and by ignoring round-trip time (RTT) probes or NAKs. However, this situation is unlikely because subgroups are usually small unless the network topology is extremely flat, i.e. a topology with a large average router fanout, or a node with low latency to most of the other nodes.

3.3. SCREAM: An Extension to OTERS

Source-Constrained-REpAir Multicast (SCREAM) restricts all traffic, including repair traffic, to come from the multicast source. That is, instead of directly subcasting the repair, the would-be retransmitter unicasts a "source-constrained-repair" request to the source. The source then subcasts the repair from the intended subroot.

SCREAM allows OTERS to function with a single-source multicasting model [7] and with a simpler encapsulation scheme, as described in Section 2.2. With conventional multicast, it allows receivers to safely ignore retransmissions from unknown nodes as well as apply source filtering to selectively disable delivery of packets, as proposed for IGMP v.3 [2]. It further prevents potentially bogus retransmissions that may come across wide area networks and whose originator's identity can be hard to verify. Otherwise, a receiver is unable to stop malicious or failed nodes from flooding the multicast group with subcast packets. Lastly, SCREAM provides better congestion control to the source because it is then well informed of losses and their locations, and thus can dictate the data rate and repair rate.

SCREAM incurs the extra delay of the round-trip to the source. However, our simulations indicate that this does not significantly increase the recovery latency. It may also lead to NAK implosion and excessive load at the source if there are losses in many independent subtrees. However, if this situation arises with any frequency, the network service and application performance is already poor, so SCREAM is unlikely to make it significantly worse for the application. For truly large scale dissemination, a few trusted regional logging servers such as in LBRM [8] can be deployed to handle the retransmission load.

SCREAM can be used in the wide area network while direct DR retransmission can be used in local networks such as corporate or campus intranet so that the retransmission need not traverse the tail circuit to the local network. Local DR retransmissions can be verified based on the source address and are generally trustworthy. This further reduces the recovery latency as well as the retransmission load for the source.

4. Simulations and Analysis

We implemented OTERS and SCREAM on NS [17] along with four other reliable multicast protocols (see also Section 5): (1) SRM [5]: multicast retransmission with duplicate avoidance (DA).[8] (2) SRM-ideal: identical to SRM but without DA, so it finds the lower bound of the recovery latency while maximizing the recovery traffic. (3) TMTP [19, 10]: using *Expanding Ring Search (ERS)* for the fusion tree and limited scope multicast. (4) TMTP-unicast: TMTP but cascaded unicast for loss recovery[9].

4.1. The Simulation Environment

The simulation is driven by a packet stream from a CBR (Constant Bit Rate) source to a multicast group. Routers run dense mode multicast routing similar to DVMRP [18]. Group members join the multicast group at time 0 and start measuring RTTs or forming the fusion tree. The data stream runs for 2 seconds starting at time 0.2 second, when the fusion tree formation is about half way through or less. This overlapping allows realistic protocol evaluations under the assumption that the network topology and group membership may change significantly every 2 seconds, which may trigger major parts of the fusion tree being rebuilt.

Twenty transit-stub network topologies are generated by GT-ITM [1]. On each of them the same simulations run 10 times with different seeds. Ten of the topologies are of 100 nodes, the other ten use 600 nodes. Ten percent of the nodes are randomly picked out as multicast group members. Links are symmetric with packet error rate of 0.5%. The transmission delay is approximately 3 ms on local links, 27 ms on links outside stub domains, and $110 \sim 150$ ms on links connecting transit domains.

4.2. Simulation Results on 100-node Topologies

Figure 3 shows the network traffic of the six protocols averaged over simulation runs. OTERS and SCREAM incur substantially lower network traffic for recovery than the other protocols, with TMTP-unicast being the closest in performance.

Figure 4 shows the end-station measured recovery latency, from the time a loss is detected to the time a repair is received. The OTERS recovery latency is close to that

[8] SRM was implemented by the original NS simulator. The DA period is chosen half deterministically and half probabilistically. The exponential backoff of NAKs is disabled to exclude this delay factor.

[9] We invented this variant of TMTP for comparison purposes.

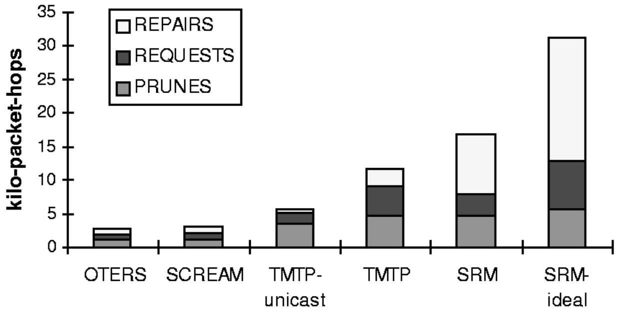

Figure 3. Network Traffic for Loss Recovery for 100-node Topology. The source data scream (not shown) is 13 kilo-packet-hops. *Packet-hops* **tracks the number of forwards by routers rather than the number of distinct packets generated by end-stations.**

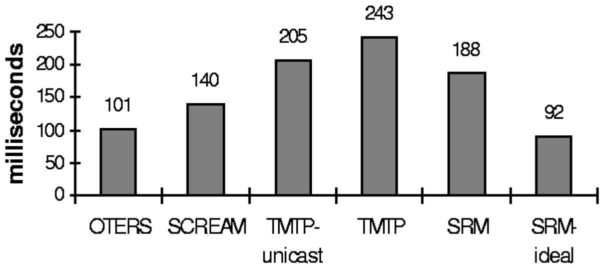

Figure 4. Average Loss Recovery Latency for 100-node Topology

of SRM-ideal, the lower bound, while having a level of recovery traffic that is far less. OTERS improves on SRM because it avoids DA delay which contributes roughly 95 ms to SRM's recovery latency. With both DA delay and cascade delay, TMTP experiences the highest recovery latency. TMTP-unicast only has cascade delay so it does better than TMTP but still far worse latency than OTERS.

4.3. Simulation Results on 600-node Topologies

A second set of simulations were run on much larger 600-node topologies to demonstrate the differences between protocols at this scale. As the multicast scale increases, differences of the recovery traffic of the six protocols grow exponentially. Figure 5 shows the network traffic on a logarithmic scale. The session traffic was added to reflect the cost of fusion tree formation and session state synchronization, negligible in 100-node topologies but significant in larger scales.

Figures 5 and 6 show that OTERS and its SCREAM variant provide even greater benefit at larger scale, improving

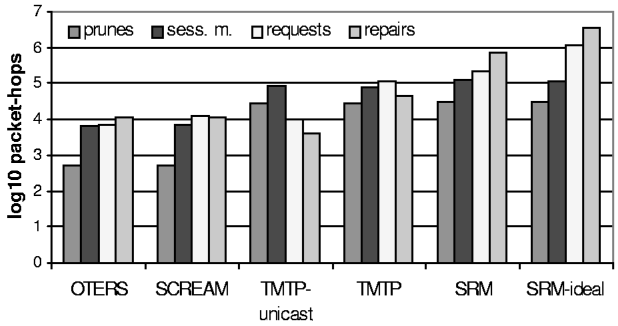

Figure 5. Network Traffic for Loss Recovery in 600-node Topology, with the addition of session messages. The source data scream is 220 kilo-packet-hops.

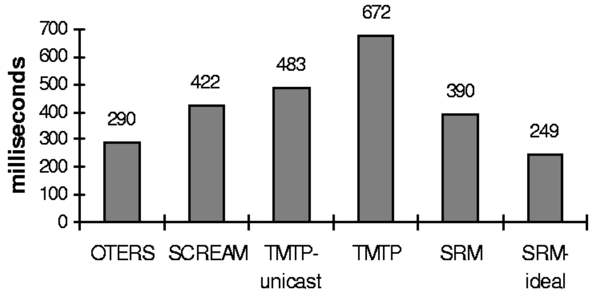

Figure 6. Average Loss Recovery Latency in 600-node Topology

by one to two orders of magnitude, even just going from 100 to 600 nodes. In particular, OTERS achieves a significantly lower recovery latency, just 17% above the lower bound while its recovery traffic is 190 times less than SRM-ideal's, 44 times less than SRM's, 10 times less than TMTP's and 5 times less than TMTP-unicast's. SCREAM also offers a competitive combination of the recovery traffic and the recovery latency. OTERS's session messages are 13 times less than TMTP's and 21 times less than SRM's.

4.4. The Early Start Effect

The early-start benefit claimed for OTERS is demonstrated in Table 1 which provides measurements from a 100-node simulation. The average recovery latency of most group members is less than the average RTT from retransmitters. For example, nodes 53 and 52 reside in the same stub network[10]. In most cases, node 52's recovery processes

[10] http://www-dsg.stanford.edu/dli/ts0.gif has the topology.

Node ID	#losses	Recovery Latency	
40	1	1.00 RTT	6.3 ms
53	5	0.77 RTT	88 ms
42	7	0.80 RTT	97 ms
52	10	0.40 RTT	50 ms
4	10	0.81 RTT	104 ms
94	10	0.81 RTT	104 ms
10	14	0.80 RTT	107 ms
14	13	0.73 RTT	98 ms
35	14	0.78 RTT	109 ms
68	13	1.17 RTT	170 ms

Table 1. Average recovery latency relative to average round-tip time for OTERS on a 100-node topology

are sped up by its upstream peer, node 53. This results in an average recovery latency as low as 0.40 RTT. On the other hand, node 68 is alone and far apart from others. It rarely benefits from a recovery process initiated by another node and therefore rarely recovers earlier than one RTT. Overall, OTERS is able to reduce the average recovery latency by topologically grouping receivers.

5. Related Work

Previous work has explored various NAK-based retransmissions schemes as well as network support for reliable multicast. Next we compare them with OTERS.

5.1 Multicast Retransmission Schemes

Table 2 summarizes the properties of five retransmission techniques including subcasting. The summary shows that subcast retransmission is faster and more efficient than conventional techniques whose operational overviews are listed below.

- *cascaded unicast* such as TMTP-unicast. Requests and repairs are unicast between a subgroup member and its DR. Correlated losses are repaired individually, generating more traffic than a multicast repair. Repairs may be retransmitted multiple times before reaching subgroups at the bottom of the fusion tree, resulting in *cascade delay*.
- *multicast with duplicate avoidance* (*DA*) such as SRM [5]. Requests and repairs are multicast to the entire group. Before transmitting a request or repair, the sender delays a DA period. The transmission is suppressed if an identical one is received during the period. Repairs are multicast to members that may not need them.
- *limited scope multicast* such as TMTP [19]. Requests are multicast with DA. Only the DR responds to requests and responds immediately. Requests and repairs are multicast to the host group with the TTL equal to the subgroup's radius to restrict the delivery scope (only in source-specific routing domains). This approach incurs both DA delay and cascade delay and suffers from the coarse granularity of TTL-scoping in the current Internet.
- *multi-group multicast*. Each subgroup has its own multicast group address. A member joins its own subgroup and also that of all its ancestor subgroups up to and including the source. Requests are unicast to the DR. The DR multicasts the repair to the multicast address corresponding to its subgroup. This approach requires many multicast addresses and the routing state for them.

Simulations in Section 4 show that these techniques incur substantially more recovery traffic while achieving recovery latencies much longer than subcasting.

Most of the above techniques use some form of hierarchical structure among group members. However, the structures used have weaker characteristics than the OTERS fusion tree.

In particular, the ERS [19, 10] structure differs because it constructs subgroups based on the closeness between group members. However, TTL- or RTT-wise closeness does not necessarily reflect their relative positions on the multicast routing tree, e.g., how much of their multicast paths is shared. If a protocol uses a fusion trees that does not model the delivery tree, correlated losses tend to scatter into more subgroups and incur more retransmissions. Also, early-start occurs less often so the average recovery latency is higher. Finally, subgroup members are more likely to receive unwanted retransmissions caused by unrelated losses in the subgroup.

RMTP [12] applied subcast retransmission, but only with statically configured retransmitters and subtree roots. The OTERS fusion tree provides automatic configuration, thereby making much larger scales practical. Moreover, RMTP did not address the issue of ensuring the originality of a subcast could be determined or that subcasting would work with the single-source multicasting model.

As demonstrated by the simulations, Subcast-based SCREAM and OTERS are less costly in network traffic than TMTP and TMTP-unicast, which are less costly than SRM because SRM requests (NAKs) and repairs are forwarded more widely than limited scope multicast. Moreover, OTERS triggers fewer DVMRP prunes than SRM, TMTP and

Protocol	Request	Repair	Network Traffic	Network State
Cascaded Unicast	individually	cascade delay	multiple unicast	zero
Multicast with DA	DA delay	DA delay	unwanted Repairs	O(# members)
Limited Scope Multicast	DA delay	cascade delay	unwanted Repairs	O(# members)
Multi-Group Multicast	early start	immediately	minimal	O(# subgroups)
Subcast	early start	immediately	minimal	zero

Table 2. The comparison of five retransmission techniques.

TMTP-unicast because each of these instances uses multiple source-specific delivery trees. Compared to them, SCREAM is particularly attractive for reliable multicast with the single-source model proposed in EXPRESS [7].

OTERS is also more efficient in session messages, as demonstrated in Section 4.3, for several reasons. First, in OTERS only DRs and new members generate FTFP subcast packets whereas every group member in TMTP and SRM multicasts session messages. Second, a TMTP subgroup-join request may be rejected by the DR, forcing the requester to try other DRs. OTERS eliminates the subgroup-join traffic and the associated delay because each group member makes unilateral decisions on who is its DR. Thus OTERS FTFP is also faster.

5.2 Network Support

Other research has explored adding router support for fast and efficient loss recovery for reliable multicast.

In the "replier-subcast" approach [13], the network forwards NAKs hop-by-hop to a replier (DR). Repairs are subcast by the replier. Unlike OTERS, it maintains an implicit fusion tree in the network layer at the cost of additional network state and protocol processing. It also requires two new IP options and a new IGMP packet format.

Second, this approach is subject to replier failures and NAK forwarding malfunctions over which the application has no direct control. Additional interactions between NAK senders and routers are necessary to overcome application-level malfunctions. Conversely, in OTERS, the application controls the fusion tree structure, making it more flexible in identifying failures and restructuring the fusion tree promptly.

Third, in "replier-subcast", NAKs are examined hop-by-hop in order to be forwarded to a replier, and therefore go through the slow path of routers. Thus it takes longer for a NAK to reach the replier than normal unicast or multicast NAKs. OTERS, on the other hand, incurs no exceptional per-hop processing overhead for NAKs and repairs. The only per-hop slow-path forwarding is the mtrace query which occurs only as part of FTFP and thus infrequently. Mtrace is also a more general-purpose facility, useful for network diagnosis and other network management applications, not just for reliable multicast.

Subcasting based on TTL[11], an alternative to our encapsulation-based subcasting scheme, seems far less attractive. Using TTL control, a subcast packet with a TTL of K is forwarded along the reverse path of the multicast route for K hops and then multicast to the subtree at the last-hop router. Similar to "replier-subcast", such subcast NAKs and repairs are forwarded hop-by-hop through the slow path of routers, inefficient for both the network and the application.

In PGM [15], NAKs are forwarded and confirmed hop-by-hop toward the source. Routers establish layer-4 retransmission soft-state for every NAK'ed segment of every multicast session and forward retransmissions only to the interfaces where NAKs have been received. The network also decides when NAKs are to be fused and when re-NAKs can be forwarded, making it harder for fine-grained application control over timing. Overall, PGM exhibits all the three disadvantages noted for the replier-subcast approach.

The "routing-label" approach [9] extends IP multicasting with rich forwarding semantics that support the delivery of packets to any specific subset of the group or subtree of the multicast routing tree. This approach is conceptually a superset of the replier-subcast approach and PGM, which implies more protocol complexity. The additional network state is computed and updated for every multicast session and on every membership change, regardless of whether the application needs it or not. Moreover, the routing labels are maintained in the network layer whereas OTERS has the end-stations cache the topology information such as the IP address of one's DR, thereby greatly simplifying the network layer and lowering the overall cost of supporting and deploying reliable multicast.

The "cache-router" approach [11] lets routers cache and retransmit dropped multicast packets. This approach consumes scarce network buffer space and also complicates the reliability of drop notifications, such as whether the router should timeout and retransmit the drop notification.

In summary, given the performance of OTERS, there seems no merit in more complex network layer support than

[11] There have been discussions of this scheme in the research community.

the simple extensions that OTERS requires. It is an open question whether there is another reliable multicast protocol that achieves performance comparable to OTERS with simpler or no network extensions.

6. Conclusion

OTERS provides reliable multicast with low recovery latency and low recovery traffic levels while requiring minimal additional network support. Simulation-based evaluation of OTERS show its recovery latency is only 1.17 times the lower bound while incurring 5 to 44 times less network traffic for recovery than other protocols and 190 times less than the protocol that produces the minimal recovery latency (Section 4.3). Moreover, OTERS can work with both single-source and multi-source multicasting, and, in the latter case, with both shared-tree and source-specific multicast routing.

A key technique of OTERS is the use of the *multicast route backtracing* facility to dynamically trace the multicast delivery tree in order to identify subroots for building a fusion tree that models the multicast delivery tree and adapts to network and membership changes.

Another key technique is the exchange of *subcast FTFP packets* among group members to distributedly select a well-positioned *designated receiver (DR)* for each subtree that handles NAK fusion and data retransmission for receivers in the subtree.

A final key technique of OTERS is using the *subcasting* facility to retransmit packets from the DR to its subtree to fast and efficiently repair packet losses in the entire subtree.

Using the above techniques, OTERS provides the best trade-off between the network performance and the end-station performance (demonstrated by the simulations), as the recovery latency and the recovery traffic are often two competing requirements in reliable multicast protocol design.

OTERS does require additional network support beyond that is present in the Internet, namely for multicast route backtracing and subcasting. However, these extensions are simple to implement, are minor modifications of existing or experimental Internet facilities, incur relatively low overhead on routers and appear to have other uses outside of OTERS and even reliable multicast. Moreover, OTERS imposes no additional state or buffering in the network layer, and requires no special per-hop processing for NAKs and repairs. Thus, these extensions fit the stateless and best-effort datagram service model of the Internet layer.

Overall, OTERS offers the potential of applications requiring low-latency reliable multicast to be deployed on the scale allowed by the Internet. We hope to see the (inter)network layer extensions for mtrace and subcasting deployed in the Internet to make this potential a reality.

References

[1] K. Calvert and E. Zegura. Gt internetwork topology models (gt-itm). *http://www.cc.gatech. edu/fac/Ellen.Zegura/gt-itm/*

[2] S. Deering, B. Cain, and A. Thyagarajan. Internet group management protocol, version 3. *Internet Draft draft-ietf-idmr-igmp-v3-00.txt work in progress*, December 1997.

[3] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei. Protocol independent multicast-sparse mode (pim-sm): Protocol specification. *RFC 2117*, June 1997.

[4] W. Fenner and S. Casner. A "traceroute" facility for ip multicast. *Internet Draft draft-ietf-idmr-traceroute-ipm-02.txt work in progress*, 1997.

[5] S. Floyd, V. Jacobson, S. M. C. Liu, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *ACM Transactions on Networking*, November 1996.

[6] M. Handley. An examination of mbone loss distributions. *http://north.east.isi.edu/mbonemon/*, March 1998.

[7] H. Holbrook and D. Cheriton. Express multicast: an extended service model for globally scalable ip multicast *http://www-dsg.stanford.EDU/holbrook/express/*, 1997.

[8] H. Holbrook, S. Singhal, and D. Cheriton. Log-based receiver-reliable multicast for distributed interactive simulation. *SIGCOMM*, 1995.

[9] B. Levine and J. Garcia-Luna-Aceves. Improving internet multicast with routing labels. *IEEE International Conference on Network Protocols*, October 1997.

[10] B. Levine, D. Lavo, and J. Garcia-Luna-Aceves. The case for concurrent reliable multicasting using shared ack trees. *Proc. ACM Multimedia'96 Conference*, 1996.

[11] M. Lim and D. Kim. Ip extension for reliable multicast. *Internet Draft draft-lim-ip-reliable-multicast-00.txt work in progress*, 1997.

[12] J. Lin and S. Paul. Rmtp: A relible multicast transport protocol. *INFOCOM*, 1996.

[13] C. Papadopoulos, G. Parulkar, and G. Varghese. An error control scheme for large-scale multicast applications. *INFOCOM*, March 1998.

[14] C. Perkins. Ip encapsulation within ip. *RFC 2003*, October 1996.

[15] T. Speakman, D. Farinacci, S. Lin, and A. Tweedly. Pretty good multicast (pgm) transport protocol specification. *Internet Draft draft-speakman-pgm-spec-00.txt work in progress*, January 1998.

[16] D. Thaler, D. Estrin, and D. Meyer. Border gateway multicast protocol (bgmp): Protocol specification. *Internet Draft draft-ietf-idmr-gum-01.txt work in progress*, October 1997.

[17] UCB, LBNL, and VINT. Network simulator - ns (version 2). *http://www-mash.cs.berkeley.edu/ns/*

[18] D. Waitzman, C. Partridge, and S. Deering. Distance vector multicast routing protocol. *RFC1075*, 1988.

[19] R. Yavatkar, J. Griffioen, and M. Sudan. A reliable dissemination protocol for interactive colla-borative applications. *Proc. the ACM Multimedia'95 Conference*, 1995.