Scalable Fair Reliable Multicast Using Active Services*

Sneha K. Kasera[1], Supratik Bhattacharyya[2], Mark Keaton[3], Diane Kiwior[3], Jim Kurose[4], Don Towsley[4] and Steve Zabele[3]

[1]Bell Labs. [2]Sprint Labs. [3]TASC Inc. [4]UMass Amherst

To appear in IEEE Network Magazine (Special Issue on Multicast), January/February 2000

Abstract

Scalability is of paramount importance in the design of reliable multicast transport protocols, and requires a careful consideration of a number of problems such as feedback implosion, retransmission scoping, distributed loss recovery, and congestion control. In this article, we present a reliable multicast architecture that invokes active services at strategic locations inside the network to comprehensively address these challenges. Active services provide the ability to quickly and efficiently recover from loss at the point of loss. They also exploit the physical hierarchy for feedback aggregation and effective retransmission scoping with minimal router support. We present two protocols, one for packet loss recovery and another for congestion control, and describe an experimental testbed where these have been implemented. Analytical and experimental results are used to demonstrate that the active services architecture improves resource usage, reduces latency for loss recovery and provides "TCP-friendly" congestion control.

1 Introduction

The ability to transmit data reliably from a sender to a group of receivers can benefit many applications such as one-to-many file transfer, information dissemination (e.g., stock quotes and web cache updates), distance learning and shared whiteboard. These applications can have potentially

*This work was done when the first two authors were Ph.D. students at University of Massachusetts Amherst thousands of receivers spanning wide area networks, hence the efficiency of data delivery is of paramount importance. Multicasting technology provides this efficiency by incurring lower network and end-system costs than traditional unicast and broadcast.

IP multicast supports efficient multicast data forwarding through the use of multicast distribution trees, but like traditional IP, it guarantees only best-effort delivery and also does not impose any restrictions on the data rate. This gives rise to two problems. First, some of the packets sent from the sender may not reach all the receivers. Second, multicast applications transmitting data at uncontrolled rates can overwhelm network resources, causing congestion, and may starve existing unicast applications of available network bandwidth. Carefully designed multicast transport mechanisms on top of IP multicast must therefore provide both reliable delivery and congestion control.

The nature of multicast communication introduces new considerations and problems in the design of scalable reliable multicast transport protocols. First, a large volume of feedback traffic may be generated by the receivers in a multicast group and may cause a feedback implosion at the multicast sender. Second, the multicast sender may have to supply a large number of retransmissions in order to ensure reliable delivery to potentially thousands of receivers. Thus the sender and the links close to it are likely to become bottlenecks and lead to degradation in overall throughput. Another problem is that packet retransmissions from the sender go to the entire group and not only those parts of the multicast tree where they are actually required. This causes unwanted packet processing at the receivers which have already received the packets and wastes network bandwidth on the links leading to these receivers.

Congestion control for large multicast groups is also a challenging problem. The congestion conditions in different parts of a multicast tree are likely to be different; these differences have to be taken into account in determining the rate of transmission of data to the entire multicast tree. For this purpose, congestion feedback must be gathered from a large number of receivers in a scalable manner. Moreover, algorithms for regulating a multicast session's transmission rate must be designed to ensure fair sharing of available network bandwidth among competing multicast and unicast sessions.

In this article, we present a reliable multicast transport architecture that uses active services at strategic locations inside the network to comprehensively address the challenges arising in large scale multicast loss recovery and congestion control. Active services advocate the placement of application level user defined computations within the network [3], while preserving the routing and forwarding semantics of current Internet architecture. They allow applications to exploit detailed knowledge of the underlying network to enhance their performance. Active services could be deployed as application level programs inside routers or, equivalently, on servers co-located with routers and could be invoked/revoked as warranted by application requirements and network conditions. In the architecture presented in this article, active services are invoked at designated servers that are co-located with routers along the physical multicast distribution trees (created by the multicast routing protocols). These active services cache packets at the designated servers to provide the ability to quickly and efficiently recover from loss at the point of loss. They also exploit the physical hierarchy for feedback aggregation and effective retransmission scoping with minimal router support. The caching of packets and subsequent loss recovery from the designated servers helps in distributing the loss recovery burden among the sender and the designated servers.

We describe two protocols, one for loss recovery and another for congestion control, and a signaling mechanism for locating and invoking active services. We also describe an experimental testbed where the two protocols and the signaling mechanism have been implemented. Next, we demonstrate the benefits of using active services in terms of reduction in network bandwidth consumption. We also demonstrate a reduction in loss recovery latency due to using active services in an actual implementation on the Active Backbone Network [1]. Finally, we use simulations to demonstrate that the congestion control protocol enables multicast sessions to "fairly" share bandwidth among themselves, and also with competing TCP sessions.

Our main contribution is in identifying a set of solutions for the problems encountered in making reliable multicast transport services scalable, and combining them into an active services framework that requires limited network support. Many of the general design ideas are not entirely novel, though some of them build on our earlier research. Nevertheless, the service architecture and the protocols that we develop provide a framework for examining these ideas in detail.

The rest of the article is organized as follows. In Section 2 we present our goals for the design of a scalable reliable multicast transport architecture. In Section 3 we present the related work on reliable multicast. In Section 4 we describe an architecture and protocols that use active services for scalable and fair reliable multicast. Section 5 demonstrates the benefits of using active services. In Section 6 we simulate the congestion control protocol and present simulation results. Conclusions and directions for future work are contained in Section 7.

2 Design Goals

Scalability, i.e. the ability to scale to large multicast groups, and efficient performance in the presence of heterogeneity in network characteristics, are two general goals for the design of any multicast protocol. In the context of reliable multicast transport, there are a number of specific issues that have to be considered in addressing these general goals:

- *Feedback implosion avoidance:* To ensure reliable transmission from the sender to the receivers it is important for the receivers to send feedback to the sender either by acknowledging every packet that is received by sending an acknowledgement message (ACK) or by sending negative acknowledgement messages (NAKs) for packets that were detected to be lost. Due to the possibility of ACK implosion and also the requirement of maintaining identity of each receiver, an ACK-based approach for reliable multicast is not scalable. A NAK-based approach is more practical because the number of NAKs is likely to be much less than the number of ACKs. Also, the sender does not need to be aware of the identity of the receivers. When the network is large and when loss probabilities are high we can incur a NAK implosion too. Avoiding NAK implosion is an extremely important requirement for scalability and should be met under all circumstances.

- *Retransmission scoping:* To ensure reliability, a lost packet must be retransmitted by the sender (or possibly some other entity). When losses are high and the number of receivers is large, unicast of retransmissions is not efficient because several identical retransmissions to many receivers might be required. In such cases, multicasting of retransmissions may be more suitable. If retransmissions are multicast, then they are sent to all receivers, not just the ones that have lost the packet. This causes unnecessary bandwidth usage on links leading to those receivers who have already received the packet, as well as unwanted packet processing at these receivers. This problem is further aggravated in the presence of heterogeneity, as a large part of the network might receive retransmissions that are needed only by a small fraction of receivers. Scoping retransmissions so that they go only to receivers which need them is an important design goal.

- *Efficient distribution of loss recovery burden:* When the number of receivers is large and/or the loss probabilities are high, the probability of at least one receiver loosing the packet is very high. This means that almost every packet needs to retransmitted and often several times. Thus the sender and the links close to the sender are likely to become bottlenecks, thereby degrading performance. Placement of processing power, to assist in loss recovery, close to the source is not very efficient in terms of network bandwidth usage and recovery latency. Thus efficient distribution of loss recovery burden is an important design goal.

- *"Fair" congestion control:* The most widely used approach in multicast congestion control is one in which a multicast source regulates its rate in response to congestion feedback from its receivers. An important challenge in designing source-based congestion control algorithms is to solicit feedback from receivers in a scalable manner, and then combine them into a single rate control decision at the source. Moreover, the rate must be regulated in a way that ensures "fair" sharing of bandwidth among competing multicast and unicast sessions.

In a design that uses active services, we have some additional goals:

- *Use active services for performance enhancements only:* The use of active services should be for performance enhancement only. The active service functionality should not be essential for correct functioning of the reliable multicast transport protocols. This goal ensures that active services could be gradually deployed in the network and that the reliable multicast protocols would continue to function correctly, although with inferior performance, even when some parts of the network do not support active services.

- *Require minimal router support:* Minimal router support should be required while preserving the routing and forwarding semantics of the current Internet architecture.

3 Related Work

In this section we review some of the existing architectures and protocols for reliable multicast transport. Most of the known reliable multicast protocols avoid NAK implosion either by suppressing NAKs using random wait [7, 19] or by aggregating NAKs using hierarchies [13, 18, 20, 22]. In order to distribute the burden of loss recovery, local recovery approaches [11, 17, 18, 22], in which entities other than the sender such as a nearby server or another receiver helps in supplying retransmissions, have been proposed. A few local recovery proposals advocate that routers [13, 20], or, servers attached to the routers [12], should cache packets and supply retransmissions if needed. This allows efficient loss recovery from the point of loss.

There have been several proposals for scoping retransmissions. Floyd *et al* [7] have discussed two schemes of scoping retransmissions, one that uses multiple multicast groups and another that is based on IP TTL (Time to Live). In addition to carefully grouping receivers, the approach that uses multiple multicast groups requires the receivers to have multicast send capability which need not be available (as explained in Section 4). TTL based scoping limits packets within a radius and is not suitable for multicast trees. Also, it is hard to approximate a good TTL value. [13, 20] have used an approach that requires routers to intercept NAKs before forwarding them towards the sender and keep NAK state for selectively forwarding retransmissions only on those links on which NAKs for those retransmissions were received. This approach promises perfect retransmission scoping but requires routers to have very fine grained control over NAKs and retransmissions. An approach that requires minimal router support, and, can be used in conjunction with local recovery, has been proposed in [14]. In this approach a multicast retransmission, from a local receiver or server, with source address set to that of the multicast sender, is encapsulated in a unicast packet and sent to the nearest router in the multicast tree. The router then multicasts the retransmission downstream on the multicast subtree (also called *subcast*) below it as if the retransmission came from the sender.

Recently there has been a proliferation of proposals for multicast congestion control. Many of these schemes (for e.g., [4, 9]) have one common underlying goal – to regulate the transmission rate of the multicast source so that bandwidth is shared fairly among multiple sessions. The source transmission rate is controlled based on aggregated receiver feedback.

Table 1 summarizes how some of the well known reliable multicast transport approaches handle

NAK implosion, loss recovery burden, retransmission scoping and congestion control. Pragmatic General Multicast, PGM, uses very fine grained router support for avoiding NAK implosion and retransmission scoping. It leaves the burden of supplying repairs on the sender. The strawman congestion control approach in PGM [20] uses NAKs for rate adjustment. It has been shown in [4] that PGM's congestion control approach scales very poorly for large multicast groups. Active

	PGM	ARM	RMTP	SRM	AER/NCA
NAK Implosion	aggregation, limited suppression	aggregation	aggregation	suppression	aggregation, suppression
Loss Recovery Burden	sender	sender, routers	sender, designated receivers	sender, receivers	sender, repair servers
Retransmission Scoping	router controlled	router controlled	router-supported subcast, local multicast groups	TTL, local multicast groups	router-supported subcast, retransmission interception
Congestion Control	reacts to every NAK at source	unspecified	reacts to losses at worst receiver	unspecified	reacts to losses at worst receiver

Table 1: Comparison of Reliable Multicast Transport Approaches

Reliable Multicast, ARM, also uses very fine grained router support for NAK aggregation and retransmission scoping. It will suffer from NAK implosion if routers do not have ARM functionality. ARM does not specify any congestion control mechanisms. Reliable Multicast Transport Protocol, RMTP, uses designated receivers for local recovery and also for NAK aggregation[1]. RMTP will

[1]It should be noted that RMTP aggregates NAKs implicitly by allowing the designated receivers to generate their own NAKs on loss detection. NAKs from the receivers are not forwarded beyond their designated receivers.

also suffer from NAK implosion in the absence of designated receivers. RMTP proposes to track the worst receiver [9] for controlling the sender transmission rate. Scalable Reliable Multicast, SRM does not require any router support, or, special servers or receivers, but does not have a satisfactory mechanism for retransmission scoping. It avoids NAK implosion through NAK suppression. For large number of receivers spanning wide area networks, NAK suppression alone is not very efficient [7]. SRM does not specify any congestion control mechanisms. We find that none of these approaches meets all our design goals. This leads us to develop an architecture and protocols (described in the next section) that use many of the ideas described in these approaches but also meet all our design goals.

4 Architecture and Protocol

We develop an architecture and protocol that effectively meets all our design goals. In this architecture active services are placed on designated servers co-located with routers. In the reliable multicast context, we call these active services *repair services* and the designated servers *repair servers.*

The repair servers have processing and buffering resources that are used to cache data for supplying repairs or retransmissions and to aggregate receiver feedback. Only those repair servers that lie along the multicast distribution tree of a multicast session are activated. Once activated, the repair servers join the multicast group on which packets are sent to the receivers and cache a "recent" set of packets which are retransmitted downstream to receivers or lower level repair servers on request. The repair servers themselves recover lost packets from upper level repair servers or the sender. The repair servers also aggregate feedback, e.g., NAKs, from downstream. Such a hierarchical architecture helps in fast local recovery from loss, reduces bandwidth usage and distributes the burden of recovery of the sender among the repair servers.

In this section we first describe a loss recovery protocol called *Active Error Recovery* (or AER in short). This is followed by a description of a signaling mechanism for locating and invoking repair services. We also identify what router support is required by AER. Next, we present a congestion control protocol called *Nominee Congestion Avoidance* (or NCA in short). We end this section with a brief description of an implementation of the architecture and protocols.

4.1 Active Error Recovery Protocol

We now present a generic version of the AER protocol. The detailed specifications can be found in [15]. The AER protocol assumes that receivers do not have multicast send capability. This is due to the following three reasons. First, in commercially available IP multicast a large amount of money is charged to the multicast sender. Second, in some networks such as the satellite networks multicast is only downstream. Third, setting up multicast distribution tree(s) for a large number of senders is expensive.

The AER protocol exhibits the following behavior:

- The sender multicasts packets to a multicast address that is subscribed to by all receivers and participating repair servers.

- On detecting a loss, a receiver (or a repair server), after waiting for a random amount of time, unicasts a NAK to its closest upstream repair server (how the identity of the nearest repair server is obtained is described in Section 4.1.1) and starts a NAK retransmission timer. If the receiver (or the repair server) receives a NAK from upstream for the same packet prior to sending the NAK, it suppresses its own.

- If a repair server has the packet for which it received a NAK from a receiver or a lower level repair server, it subcasts the packet downstream. If the repair server does not have the packet, it immediately subcasts (multicast on the subtree below the router connected to the repair server) the NAK downstream and begins the process of obtaining the packet from the sender (or its upstream repair server) if it has not already done so (as described in the previous item). In AER, a subcast NAK goes only to the next downstream repair servers (or receivers, if there are no downstream repair servers). A subcast repair packet is also intercepted by the next downstream repair servers and cached but forwarded further down the tree only if there is a pending NAK for that repair.

- The expiration of the NAK retransmission timer at a repair server, (or a receiver) without prior reception of the corresponding repair, serves as the detection of a lost packet for the repair server (or the receiver) and a NAK is retransmitted.

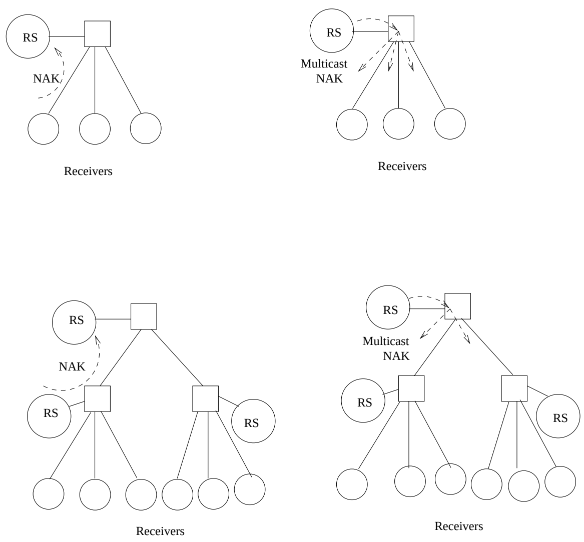

Figure 1: NAK Suppression Scenarios

- On receiving a NAK from a repair server, the sender remulticasts the requested packet to all the receivers and repair servers. As mentioned above, these repairs are intercepted by intermediate repair servers and forwarded further down the tree only if there is pending NAK for that repair.

Figure 1 exhibits two NAK suppression scenarios. Note that NAKs are unicast upstream but multicast downstream.

In summary, the AER protocol addresses the challenge of NAK implosion by using random timer-based NAK suppression and by using the repair server hierarchy for NAK aggregation. Even though NAKs can be aggregated by repair servers, NAK suppression is still used to ensure that NAK implosion does not happen when some part or all of the distribution tree is without repair servers. AER distributes the burden of loss recovery among the repair servers. It scopes retransmissions with the help of subcasting and interception of retransmissions.

4.1.1 Signaling Mechanism

In this section we present a simple signaling mechanism for the following three purposes; locating repair services, invoking/revoking repair services, and handling route asymmetry and route changes. The signaling mechanism can be described as follows.

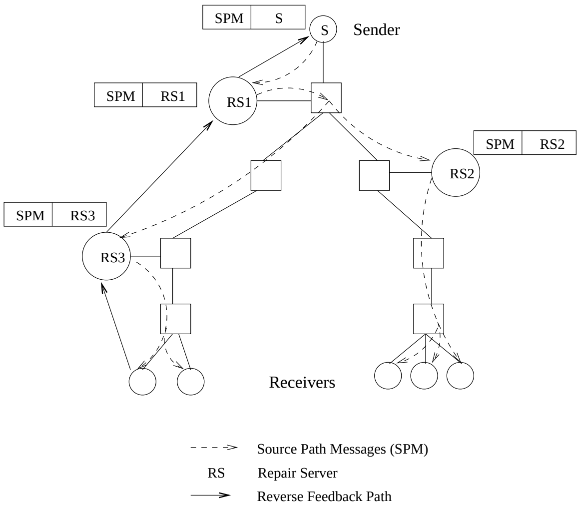

Figure 2: Reverse Path Establishment

- The sender periodically multicasts "source path messages" or SPMs in short (similar to source path messages of PGM [20] and path messages of RSVP [23]) on the same multicast group on which the data is sent to the receivers. Each SPM contains a field for storing the originator's IP address (which we will see shortly is not typically the sender's IP address).

- SPMs are intercepted by routers, with attached repair servers, along the path from the sender to the receivers and sent to the attached repair server. The router support required for interception and forwarding of SPMs to the attached repair server is described in Section 4.1.2.

- On receiving an SPM, a repair server notes down the IP address of the previous repair server or the sender. This value is carried in the SPM. A repair server then replaces that address by its own address and subcasts the SPM downstream.

- On receiving an SPM, a receiver records the IP address of the repair server in the SPM.

The above signaling mechanism establishes reverse paths from the receivers to the sender through the repair servers. Establishing reverse paths is essential because the multicast data path from the sender to a receiver could be different from the unicast data path from the receiver to the sender on which NAKs are sent. In order to to be able to send NAKs to recover lost packets, a receiver or a repair server needs to know the identity of the nearest upstream (*w.r.t.* itself) repair server that is on the multicast data path. Once the reverse path is established, a receiver or a repair server can send NAKs for lost packets to its nearest upstream repair server (which could be the sender). Figure 2 demonstrates the establishment of the reverse path from the receivers to the sender using SPMs.

Repair servers advertise themselves through the SPMs by including their IP address in the SPMs before subcasting the SPMs down the multicast tree. If for some reason a repair server does not wish to advertise itself, or wishes to unadvertise itself after having advertised itself earlier, it simply subcasts the SPM it receives without including its own IP address in the SPM.

SPMs are sent periodically from the sender to deal with route changes. Subsequent to a route change, the SPMs are sent on a different multicast tree. The new multicast tree involves new repair servers. Once again these new repair servers advertise their presence in the SPMs and the repair servers and the receivers update the address of their nearest repair servers. This allows establishment of new reverse paths from the receivers to the sender through the repair servers.

SPMs are also used for invoking repair services at repair servers along the path from the sender to the receivers. Those repair servers which advertise themselves in the SPMs, also start repair services. When there is a route change, no SPMs are received by the repair servers along the old path. These repair servers, after waiting for a certain specified amount of time, revoke the repair services and free up all the resources that are used by the repair services.

In summary, the signaling mechanism presented above provides a very simple way of locating, invoking/revoking repair services while flexibly handling route asymmetry and route changes.

4.1.2 Router Support

We now look at the new functionality required at routers to support AER and the above signaling mechanism. New functionality is needed at the routers to support subcast and interception of SPMs, repairs, and NAKs.

One possible approach for achieving subcast and the interception of SPMs, repairs, and NAKs is to use a separate multicast group, for each repair server, which all the next level downstream repair servers (or receivers) must join. Subcast packets (SPMs, repairs, and NAKs) are sent by repair servers on these multicast groups and received by the next level downstream repair servers or receiver. Even though this approach does not require any additional router support, it requires a large number of multicast groups and their management and hence is not practical. We take a different approach in which the following simple functionality for subcast and interception of packets could be added at a router without requiring any changes in its forwarding mechanism.

Subcast support: Subcast can be achieved using IP encapsulation [14] as follows. A subcast packet is sent unicast to the router co-located with the repair server. This unicast packet encapsulates an IP packet header that contains the IP address of the original multicast sender in the source address field and the multicast group address in the destination address field. On receiving the unicast from the repair server, the co-located router multicasts the encapsulate packet on the multicast subtree below it as if it was from the original sender. The router needs to provide the functionality to multicast the encapsulated packet downstream.

Interception of SPMs, NAKs, and Repairs: A router must distinguish between packets that need attention of the co-located repair servers such as the SPMs, NAKs and repairs, and the packets that need to be simply forwarded. The IP protocol header allows certain optional fields that can be used in an IP packet to get per hop attention as it is forwarded from the packet source to destination. These optional fields are called IP options. A *router alert* IP option can be used in a packet to "get the attention" of a router. The routers that recognize this option will send the packet containing this option to the co-located repair server. Routers that do not recognize this option or do not have any co-located repair servers simply forward this packet.

SPMs sent on a multicast group need to be intercepted at routers with attached repair servers even before repair services are invoked and before the repair server has joined the multicast group. We define a new IP router alert option called an SPM option. When a router co-located with a repair server receives an IP packet with the SPM option, it forwards the packet to the repair server. The router needs to maintain the address of the outgoing interface leading to the repair server.

The AER protocol requires subcast NAKs and repairs to be intercepted at the next downstream routers that have co-located repair servers and sent to the repair servers. We define another IP router alert option called Repair-NAK that is included in the repairs and the NAKs and is used for identifying a repair or NAK as a packet requiring special attention at a router with co-located repair server. The router with a co-located repair server gives special attention to a packet with the Repair-NAK option but unlike in the case of SPMs it intercepts and sends the packet to the repair server only if the repair server is a member of the multicast group on which the NAKs and repairs are sent. If the repair server is not a group member, then the router simply forwards the packet downstream.

4.2 Nominee-Based Congestion Control

We now describe the congestion control protocol designed to operate with the AER loss recovery protocol. The protocol, named *Nominee-based Congestion Avoidance* (NCA), uses a source-based rate adjustment algorithm that regulates the transmission rate according to packet loss indications from a single receiver – the *nominee* – in the multicast group. In this article, we provide a functional description of NCA. The detailed specifications are available in [15].

4.2.1 Nominee Selection

An important goal for the NCA protocol is "TCP-friendliness", which mandates that a multicast session must not receive more bandwidth than competing TCP sessions on any of the source-to-destination paths in the multicast tree. In order to achieve TCP-friendliness, the worst path in a multicast tree is determined as the path on which a TCP session will receive the least bandwidth. If the effect of timeouts is ignored, the the average throughput (bandwidth) of a TCP session with fixed-size packets is given by

$$B(p,T) = C/(T * \sqrt{p}) \tag{4.1}$$

where p is the loss probability estimate and T is the round-trip time estimate to the source for a receiver, and C is a constant [6].

Hence finding the *most bandwidth constrained path* in the TCP sense corresponds to finding the path with the highest value of the function $g(p,T)$, given as

$$g(p,T) = T * \sqrt{p} \tag{4.2}$$

The NCA protocol determines the worst-path as the one with the highest value of the function g, and nominates the multicast receiver at the end of this path for soliciting per-packet acknowledgments.

A nominee is selected as follows:

- Each receiver, estimates its end-to-end packet loss probability, p, using a fixed size sliding window, and, the round trip time, T, from the multicast source to the receiver. It periodically unicasts the values p and T in a Congestion Status Message (CSM) to its closest upstream repair server.
- Based on periodic CSMs from all of its downstream repair servers/receivers, a repair server identifies the "worst" receiver as the one with the highest value of the function g, given by Equation (4.2). It then unicasts the CSM of the worst receiver to its nearest upstream repair server.
- The source also receives periodic CSMs from its downstream repair servers and receivers. It uses the same method as a repair server to select the worst receiver downstream from it, which is also the worst receiver in the entire multicast group.
- Once the source has identified the worst receiver or "nominee", it unicasts a message to this receiver soliciting per-packet acknowledgments from it.

4.2.2 Rate Adjustment Algorithm

The rate adjustment algorithm determines how the source adjusts its rate in response to per-packet acknowledgments (ACK) received from the nominee. The source maintains a congestion window, W, and a slow-start threshold variable $Thresh$. W is adjusted in a manner very similar to TCP New Reno [8], though there are some differences. Briefly, the congestion window is adjusted as follows :

On receiving ACK, if ($W < Thresh$) $W \rightarrow W + 1$.
else $W \rightarrow W + 1/W$.
On detecting loss, $Thresh \rightarrow W/2$; $W \rightarrow W/2$ (congestion avoidance).
On timeout, $Thresh \rightarrow W/2$; $W \rightarrow 1$.(slow start).

A packet is assumed to be lost if the source receives ACKs for three packets that were transmitted after this packet. The source performs fast recovery when it detects a loss, but it never reacts to more than in one loss per transmission window [8]. A timeout occurs when the source does not receive any ACKs for a certain time interval from the time of transmission of the most recently transmitted packet. The timeout interval is based on a smoothed estimate of the round-trip time, which is computed using timestamps in data packets and ACKs [15].

In summary, we observed that NCA satisfies all our design goals for source-based multicast congestion control algorithms. It enables fair bandwidth sharing among competing sessions, and prevents both multicast and unicast sessions from being starved of bandwidth. It is TCP-friendly, thereby making it suitable for deployment in today's Internet where TCP is the dominant transport protocol by far. Scalability is accomplished in two ways; first, the "worst path" approach ensures that multicast throughput degradation is not drastic for large groups [4]. Second, repair servers are used to solicit congestion feedback from receivers in a scalable manner. Finally, NCA has been designed to operate independent of the protocol used to repair packet losses. This modularity of design allows loss repair and congestion control to be offered as separate services within a reliable multicast transport architecture.

4.3 Implementation

The testbed implementation of AER and NCA uses the Active Node Transfer System (ANTS) from MIT [21]. ANTS is an active networking execution environment implemented in Java. It provides a node runtime that can participate in an active network, and a protocol programming model that allows users to customize the forwarding of their packets. It does this using the concept of a "capsule," where each capsule contains not only the data to be transferred, but also a reference to the code that should be used to process the capsule at each ANTS node. ANTS also handles transferring the Java class files required for a capsule from one node to the next until all the nodes in the path have the code required to process the capsule. Subsequent capsules of the same type use the cached code, enhancing the capsule processing speed.

ANTS utilizes UDP/IP for transferring capsules between nodes. Each capsule uses an Active Network Encapsulation Protocol (ANEP) header [2] for identifying ANTS as the capsule's execution environment, allowing the resulting protocol to function over the ABone [1]. At each ANTS node, a static routing table is used that is set up before the node is started. Only the ANTS routing and

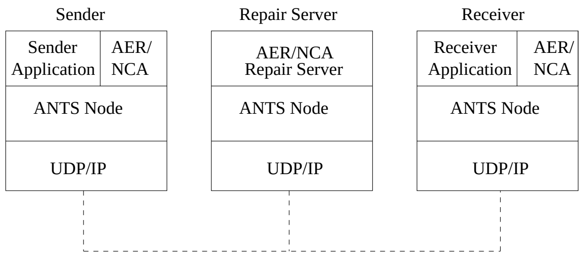

Figure 3: Block Diagram of the Implementation

caching services are used in the implementation. A block diagram of the resulting implementation is shown in Figure 3. It is important to note that we need to run ANTS at the end-hosts (sender and receivers) as well as in the network in order to route capsules into the ANTS topology.

At the repair server, the AER/NCA repair server object is created by the arrival of the first SPM capsule at the node. This object is then placed in the ANTS node cache to be used by all subsequent AER/NCA capsules of all types.

There is a very important difference in our testbed implementation using ANTS and the design of our reliable multicast architecture. Our architecture uses active services with minimal router support whereas ANTS provides an active networking environment which allows completely programmable networks where individual packets can be programmed to take arbitrary actions as they propagate through the network. We chose the ANTS environment mainly because it provides flexible deployment of protocol code and also because of its compatibility with ABone.

5 Benefits of Active Repair Services

In this section we establish the performance benefits of using active repair services. We use the following two performance measures.

- Network bandwidth usage - The network bandwidth usage is defined to be the sum of all the bytes transmitted on all the links in the multicast distribution tree for reliable transmission of a packet from the sender to all the receivers.
- Repair Latency - The time elapsed between the first detection of a packet loss at a receiver and the receipt of first repair for that packet.

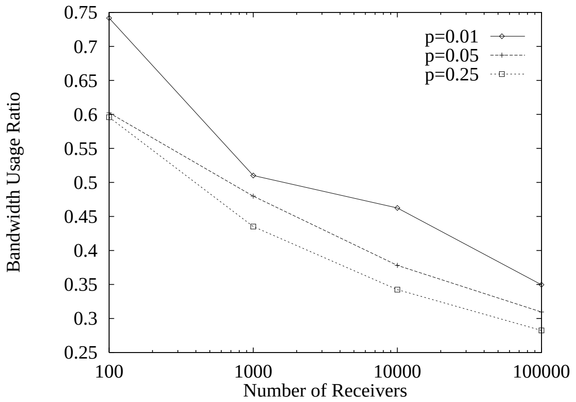

Figure 4: Bandwidth Reduction (active repair service turned on/no repair service)

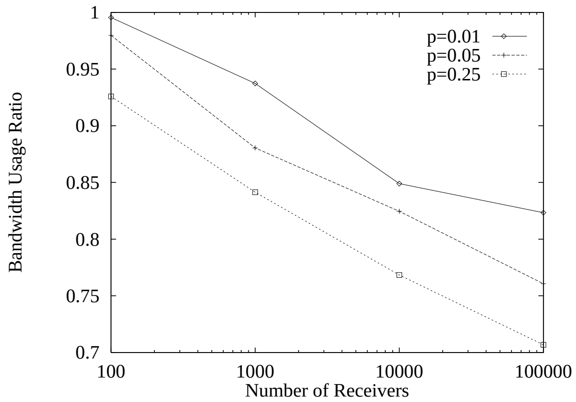

Figure 5: Bandwidth Reduction (active repair service/repair service in receiver domain)

5.1 Network Bandwidth Usage Reduction

We use the models and analyses of [12] to make the network bandwidth usage comparison. We assume that losses takes place only on the link(s) between the multicast sender and the backbone network and the links between the backbone network and the receivers. Active repair services are invoked at repair servers co-located with routers at the edge of the backbone. We first compare two scenarios under AER, one in which active repair services are turned on and the other in which they are turned off.

Figure 4 shows the relative performance of AER with active repair services are turned on and with active repair service turned off. In this figure p is the probability of end-to-end loss seen by each receiver. We observe that there is a 24% to 72% reduction in network bandwidth usage. This reduction in network bandwidth usage can be attributed to the local recovery of losses when the active repair services are turned on.

In order to highlight the performance improvement due to invoking repair service inside the network, we compare the network bandwidth usage in the following two scenarios. In the first scenario, we use AER with active repair services invoked at repair servers placed at edge of the backbone and in the second scenario we use a similar [12] protocol with repair services placed in the receiver domain.

Figure 5 shows how the network bandwidth usage reduces under AER in comparison to the other protocol with increase in the loss probability (p) and the number of receivers. This behavior can

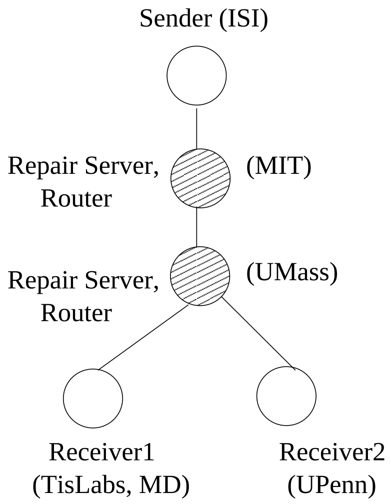

Figure 6: Multicast Topology

be attributed to the fact that placement of repair services inside the network above loss, compared to placement of repair service in the receiver domain, allows a less lossy and shorter recovery path. Figures 4 and 5 also highlight the scalability of the active repair services approach. It is important to note that the placement of the designated receivers [17] at strategic locations inside the network in RMTP (or other equivalent approaches) will produce similar performance benefits.

5.2 Repair Latency Reduction

In order to verify the benefits of using active repair services in a real-world setting we conduct experiments by running the AER/NCA testbed on the Active Backbone Network, ABone [1]. The ABone is a virtual network of computers that support active networking. Each computer on the ABone allows registered clients to execute a set of control commands. These control commands can be used to download and execute active network application code at remote machines.

In this section we present the results of the AER/NCA implementation using the topology shown in Figure 6. We run two reliable multicast sessions concurrently, one with active repair services turned on and the other with active services turned off. One kilobyte packets are sent at an NCA controlled rate for 300 seconds for both the sessions. Table 2 summarizes the repair latency (in milliseconds) results of the experiment. We observe that there is a significant reduction in mean repair latency when active repair services are turned on. From the last two columns we also observe

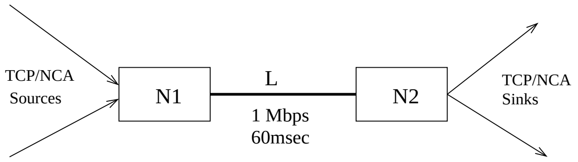

Figure 7: Simulation Configuration A - completely correlated losses

that when active repair services are turned on the mean repair latency is a small fraction of the mean round trip time and when the active repair services are turned off it is almost the mean round trip time as expected.

The reduction in mean latency is due to the following two reasons. First, active repair services provide local recovery. Second, the repair servers in AER, on detecting loss, pro-actively fetch lost packets.

	ML1	ML0	ML1/ML0	ML1/RT	ML0/RT
Receiver1	95.65	264.84	0.36	0.37	0.99
Receiver2	63.39	265.32	0.24	0.27	1.04

Table 2: ML1 - Mean Repair Latency with repair service turned on, ML0 - Mean Repair Latency with repair service turned off, RT - Mean Round Trip Time from Receiver(1 or 2) to the sender

6 NCA Simulations

In this section, we present simulation results that demonstrates that the NCA protocol enables multiple multicast sessions to share available network bandwidth "fairly" among themselves, and also with competing TCP sessions. As mentioned earlier, we adopt the "worst-path" approach where a multicast session's transmission rate is determined by the conditions on the worst path in the multicast tree. On this path, the multicast session is allocated as much bandwidth as a competing TCP session that traverses this path.

In the simulation study, we focus on only the bandwidth sharing aspect of NCA. We limit ourselves to small multicast groups, that are sufficient for illustrating TCP-friendliness, but not so for a

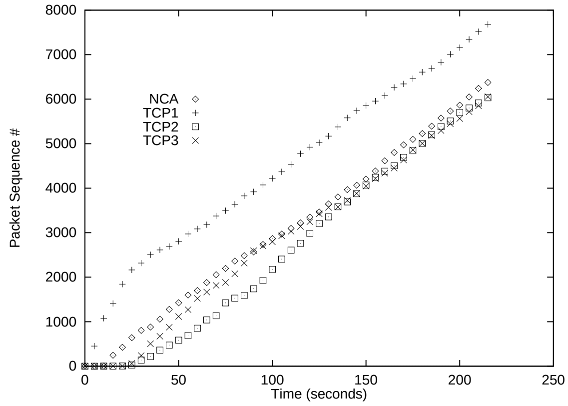

Figure 8: Configuration A simulation 1 : one NCA session competing with three TCP sessions.

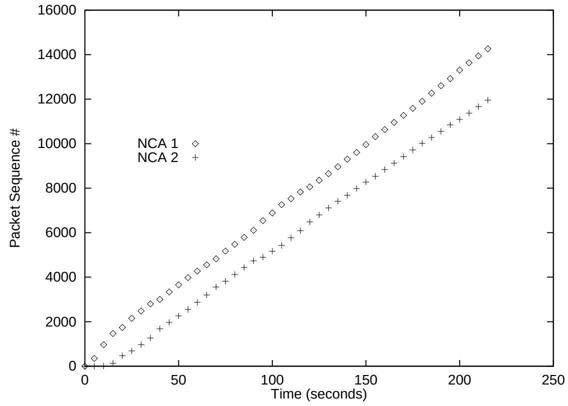

Figure 9: Configuration A simulation 2 : two competing NCA sessions

non-trivial demonstration of the benefits of feedback aggregation. A demonstration of this benefit through simulations of very large multicast groups is left as future work.

The LBNL ns simulator is used for simulating a number of simple multicast topologies. These topologies have been suggested in [10] as reference topologies for reliable multicast congestion control. Every session, unicast or multicast, has an infinite data source; every unicast session uses the TCP New Reno congestion control algorithm, while every multicast session uses NCA. All sessions use packets of size 1 Kbytes. The bandwidth share of a session is measured in terms of the mean transmission rate r, which is defined as follows. If the session's source transmits b packets in the interval $[t_1, t_2]$, then $r = b/(t_2 - t_1)$.

The first set of simulations are for Configuration A (Figure 7), where there are multiple sessions traversing a single link L with a bandwidth of 125 packets/sec, a propagation delay of 60 msecs, and a buffer size of 30 packets, connecting nodes $N1$ and $N2$. In simulation 1, there is one multicast sessions using the NCA protocol with the source attached to $N1$ and two receivers attached to $N2$. In addition, there are three TCP-Reno sessions, TCP1, TCP2 and TCP3 with the sources attached to $N1$ and the sinks attached to $N2$. The simulation is run for 220 seconds, with the NCA session starting at time 10.0, and the TCP sessions TCP1, TCP2 and TCP3 starting at times 1.15, 21.75 and 21.43 seconds respectively. In this configuration, the multicast session has two equally congested receivers; NCA chooses one as its nominee and regulates its rate according to

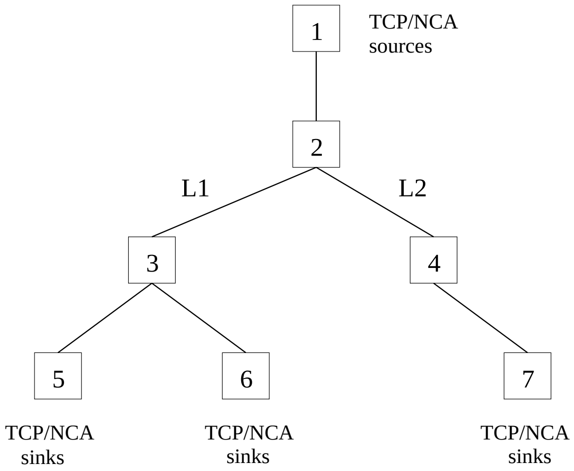

Figure 10: Simulation Configuration B - uncorrelated loss

the rate adjustment algorithm described in Section 4. From Figure 8, we observe that the NCA session and the three TCP session receive approximately the same share of link bandwidth on link L, illustrating the TCP-friendliness of NCA. We also observe that the NCA protocol is able to converge fairly quickly to its steady state when there is a change in the network congestion level, caused by the introduction of TCP2 and TCP3. We have conducted the same simulation with upto 72 receivers for the multicast session, and have observed the same behavior.

In simulation 2 for Configuration A, there are two multicast sessions over L, started 10 seconds apart, each having two receivers attached to $N2$. We observe from Figure 9 that the mean transmission rate for both sessions is approximately the same, indicating that multiple multicast sessions using NCA share bandwidth fairly among themselves.

In order to explore the effect of spatially uncorrelated loss on the behavior of NCA, we next consider Configuration B (Figure 10). In this configuration, there is a single multicast session with the source at node 1 and receivers at nodes $5, 6$ and 7. Two simulations are conducted, varying the number of TCP sessions that compete with the multicast session on different source-to-destination paths in its multicast tree. The links $L2$ and $L3$ have bandwidths of 200 packets/sec each, while all of the other links are much faster at 1000 packets/sec.

In simulation 1, there is a TCP session TCP1 from node 0 to node 5, TCP2 from node 0 to node 6, and TCP3 and TCP4 from node 0 to node 7. In this case, the bottleneck links $L1$ and $L2$

are equally loaded, and we find that the mean transmission rate of the multicast session was 62.5 packets/sec, while those for the TCP sessions are 67.5, 66.3, 65 and 67.5 packets/sec respectively. In other words, the multicast session receives almost an equal share of bandwidth as any of the TCP sessions traversing $L1$ or $L2$. This shows that even when packet losses are spatially uncorrelated, and there are equally congested links in the multicast tree, NCA behaves in a "TCP-friendly" manner towards all TCP sessions that compete with it for available bandwidth.

Simulation 2 has four TCP sessions from node 0 to node 5, four from node 0 to node 6, and two from node 0 to node 7; everything else is unchanged from simulation 1. In this case, $L1$, with 9 sessions traversing it, is more congested than $L2$ which has 3. In this case, we find that the throughput of the NCA session is 26.4 packets/sec, about the same as the average of the throughputs of the TCP sessions traversing $L1$ (21.375 packets/sec). The TCP sessions on $L2$ receive more bandwidth on an average (84.9 packets/sec) since they traverse a less congested path. This illustrates that NCA is friendly to TCP sessions on every source-to-destination path in a multicast tree.

7 Conclusions

In this article we have presented the design, evaluation and implementation of a reliable multicast transport architecture and protocols that use active services to comprehensively address the issues of scalability and fairness that arise in large scale multicast loss recovery and congestion control. We established that the use of these active services results in higher protocol performance in terms of reduced network bandwidth usage and reduced repair latency. Our simulation results demonstrated that the congestion control protocol allows fair sharing of available network bandwidth among competing multicast and unicast sessions.

We have presented a very simple approach for invoking and revoking active services. In this approach repair services are invoked by the source path messages and get revoked if no source path messages are received for a long time. There is no provision of revoking repair services based on changing network conditions, for example network loss. Another limitation of our architecture is that the repair services are embedded in the AER/NCA protocols. An important future work would be to design an architecture that makes the active services available as composable services which can be invoked/revoked by different protocols and applications as warranted by their requirements and network conditions.

We have implemented the AER and NCA protocols on the ANTS platform. In the future we would like to test these protocols on other platforms too (for e.g., CANES [5] from Georgia Tech). An issue under investigation for NCA is how to respond in a timely fashion to changes in network congestion conditions. Such changes require the selection and activation of a new nominee by the source, and a re-initialization of the window that is maintained for the rate adjustment algorithm. The challenge here is to restore steady state conditions quickly without perturbing network state significantly during the transient phase.

Acknowledgements

The authors would like to thank Brian DeCleene, Timur Friedman and Dan Rubenstein for their helpful suggestions.

References

[1] Active Network Backbone. http://www.isi.edu/abone.

[2] D.S. Alexender, R. Braden, C.A. Gunter, A.W. Jackson, A.D. Keromytis, G.J. Minden and D. Wetherall, *Active Network Encapsulation Protocol (ANEP)*, RFC Draft, July 1997.

[3] E. Amir, S. Mccanne and R. Katz, *An Active Service Framework and its Application to Real Time Multimedia Transcoding*, Proceedings of ACM Sigcomm Conference, September 1998.

[4] S. Bhattacharyya, Jim Kurose and Don Towsley, *The Lost Path Multiplicity Problem in Multicast Congestion Control.* Proceedings of IEEE Infocom, March 1999.

[5] CANES: Composable Active Network Elements. http://www.cc.gatech.edu/projects/canes/

[6] S. Floyd and K. Fall. *Promoting the Use of End-to-end Congestion Control in the Internet.* IEEE/ACM Transactions on Networking, August, 1999.

[7] S. Floyd, V. Jacobson, S. McCanne, C. Liu and L. Zhang, *A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing.* IEEE/ACM Transactions on Networking, Vol.5, No.6, pages 784-803, December 1997.

[8] S. Floyd and T. Henderson, *The NewReno Modification to TCP's Fast Recovery Algorithm.* RFC 2582, Experimental, April 1999.

[9] J. Golestani and K. Sabnani, *Fundamental Observations on Multicast Congestion Control in the Internet.* Proceedings of IEEE Infocom, March 1999.

[10] M. Handley, *Simulations for Reliable Multicast Congestion Control Schemes.* Presentation at the IETF RMRG Workshop, London, UK, 1998. http://www.east.isi.edu/rm/london/mgh.ps.gz.

[11] M. Hofmann, *Enabling Group Communication in Global Networks.* Proceedings of Global Networking, November 1996.

[12] S. Kasera, J. Kurose and D. Towsley, *A Comparison of Server-Based and Receiver-Based Local Recovery Approaches for Scalable Reliable Multicast*, UMass Tech Report, October 1998 (an earlier version appeared in Proceedings of IEEE Infocom, March 1998).

[13] L. Lehman, S. Garland and D. Tennenhouse, *Active Reliable Multicast.* Proceedings of IEEE Infocom, March 1998.

[14] J. Lin and S. Paul, *RMTP: A Reliable Multicast Transport Protocol.* Proceedings of IEEE Infocom, 1996.

[15] PANAMA Project. http://www.tascnets.com/panama.

[16] C. Papadopoulos, G. Parulkar and G. Varghese, *An Error Control Scheme for Large-Scale Multicast Applications.* Proceedings of IEEE Infocom, March 1998.

[17] S. Paul, J. Lin, K. Sabnani and S. Bhattacharyya, *RMTP: A Reliable Multicast Transport Protocol.* IEEE Journal on Selected Areas in Communications,Vol.15, No. 3, pages 407-421, April 1997.

[18] S. Paul, K.K. Sabnani and D.M. Kristol, *Multicast Transport Protocols for High Speed Networks.* Proceedings of ICNP, 1994.

[19] S. Ramakrishnan and B.N. Jain, *A Negative Acknowledgement with Periodic Polling Protocol for Multicast over LANs.* Proceedings of IEEE Infocom, March 1987.

[20] T. Speakman, D. Farinacci, S. Lin and A. Tweedly, *Pragmatic General Multicast.* Internet Draft, August 1998.

[21] D.J. Wetherall, J. Guttag and D.L. Tennenhouse, *ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols.* Proceedings of IEEE OPENARCH, April 1998.

[22] R. Yavatkar, J. Griffioen and M. Sudan, *A Reliable Dissemination Protocol for Interactive Collaborative Applications.* Proceeding of ACM Multimedia, November 1995.

[23] L. Zhang, S. Deering, D. Estrin, S. Shenker and D. Zappala, *RSVP: A New Resource Reservation Protocol.* IEEE Network Magazine, September 1993.