A Third-Party Value-Added Network Service Approach to Reliable Multicast

Kunwadee Sripanidkulchai Andy Myers Hui Zhang

Carnegie Mellon University
{kunwadee,acm,hzhang}@cs.cmu.edu

Abstract

We propose a third-party value-added services framework for enhancing the performance of reliable multicast protocols. In this framework, a value-added service provider will place servers called *waypoints* throughout ISPs' networks. Waypoints run a fully distributed, dynamic algorithm to determine which multicast groups to join. Having joined a group, a waypoint participates in the error recovery protocol, supplying repairs to receivers. From the application's perspective, waypoints appear to be additional application endpoints in the network. Waypoints seamlessly interoperate with current reliable multicast algorithms with only a minor change to receivers and no changes to routers. In our implementation, receivers and waypoints use STORM, a structure-based error recovery protocol. The waypoint recovery service is not limited to one error recovery protocol, and can be extended to enhance other reliable multicast protocols. Results from simulation experiments are presented to evaluate the potential benefits of theproposed scheme. We find that when multicast group members are isolated from each other, a waypoint recovery service can significantly enhance receivers' performance.

1 Introduction

While traditional network and transport protocols support only point-to-point communication services, there has been a growing number of network applications that require support for multipoint communication. Examples include video, audio, and whiteboard conferencing; distributed simulation; and news dissemination. IP Multicast provides a powerful abstraction and is an efficient mechanism at the network layer. However, it is only a best-effort service, which is insufficient for applications requiring reliability.

A variety of approaches to support reliable multicast have been proposed. The challenge is to achieve scalability with a large number of receivers in heterogeneous environments. In traditional unicast transport protocols, such as TCP, the sender is responsible for both loss detection and packet retransmission. In reliable multicast protocols, such a sender-based scheme means that the sender needs to perform these functionalities for every receiver, which will make the sender a bottleneck in the presence of a large receiver set. Therefore, the key to achieving scalability in reliable multicast is to *distribute* the functionalities of error detection and recovery to entities other than the sender. Existing solutions can be classified into three categories, depending on which functionalities are distributed to which entities.

In receiver-based protocols [5, 6, 11, 19, 22, 23, 24], all application endpoints, including both senders and receivers, cooperate to detect and recover packet losses. The performance of these protocols usually varies depending on the topological distribution of senders/receivers, correlation of errors, and effectiveness of underlying IP Multicast support (whether IP Multicast can efficiently support a large number of groups or frequent membership changes, whether IP Multicast has a good scoping mechanism, and whether all receivers can also be senders). Some protocols [1, 4] explicitly take differences in receiver performance into account by sorting receivers into groups with similar loss characteristics.

In router-based solutions, routers are modified to assist application end points with error detection and recovery. Solutions vary from adding minimum support in the IP layer [10, 12, 16, 20], to assuming an active network infrastructure [9]. Router-based protocols usually achieve better performance than receiver-based protocols for several reasons: routers are ubiquitous, they are located on the data distribution path, and they have access to routing information. However, overloading router functionalities introduces both scalability concerns and deployment barriers.

A third class of reliable multicast protocols utilize specialized servers to help with error detection and recovery [8, 17]. Unlike router-based schemes, server-based protocols do not necessarily need special router support for reliable multicast. The key to achieving good performance in server-based schemes is placing servers in strategic locations, so they can help detect and recover errors faster than ordinary receivers. Most existing server-based protocols are designed for applications (e.g. financial information distribution to long-term subscribers) with predetermined, semi-static group membership in a private network environment. With these protocols, the servers are usually *statically* placed at strategic locations and receivers are *a priori* assigned to servers. These protocols usually do not work well in a dynamic environment where groups and group memberships are not known a priori.

In this paper, we propose a third-party value-added ser-

This research was sponsored by DARPA under contract numbers N66001-96-C-8528 and E30602-97-2-0287, and by NSF under grant numbers Career Award NCR-9624979 and ANI-9814929. Additional support was provided by Intel Corp. Views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of DARPA, NSF, Intel, or the U.S. government.

SIGMETRICS '99 5/99 Atlanta, Georgia, USA

vice approach to support reliable multicast in an internetworking environment containing multiple service providers. We consider a network model in which there are three entities: applications, value-added service providers (VASP), and the underlying bitway IP network. Multi-point applications communicate with each other using a reliable multicast protocol that does not need to assume router or server support, i.e. a receiver-based protocol. A reliable multicast service VASP will place *waypoint* [1] machines (machines providing the recovery service) throughout the Internet on different ISPs. A distributed algorithm will dynamically bind waypoints to multicast groups and receivers to help with the error recovery. Once a waypoint decides to help a reliable multicast session, it will join the group and run the same protocol as if it were an application endpoint.

From the application's point of view, these waypoint machines are part of the network, and provide a value-added service by packaging storage and computation resources at strategic locations. While traditional applications rely on the computation and storage capabilities of application endpoints and the communication service provided by the network, with value-added service providers an application can also leverage storage and computation resources beyond its own endpoints. On the other hand, from the network's point of view, these waypoint machines are ordinary end systems, and not part of the IP infrastructure. Therefore, there is no need to change the routers.

While there are many receiver-based protocols, in this paper we consider a VASP-based multicast error recovery scheme for applications that run the STORM[23] error recovery protocol. Through simulation, we evaluate the effectiveness of waypoints in helping receivers to recover lost packets. We demonstrate that waypoints are extremely beneficial in common situations where receivers alone are ineffective, both in terms of recovering lost packets and also in reducing the burden from participating in the recovery protocol on receivers. Adding waypoints to the original multicast data distribution tree increases the multicast bandwidth in the network. However, waypoints only provide service as needed by receivers and should impose only a moderate amount of additional load on the network. This study demonstrates that value-added services can be both effective and efficient, offering a marked performance improvement for applications without overloading the network. In this paper, we focus on the potential benefit of waypoints, leaving the task of fine tuning the system for future work.

In the next section, we begin the description of our system's architecture by reviewing STORM, the error recovery protocol used by receivers. In Section 3, we continue by describing the interactions between waypoints and receivers, including how waypoints determine when to begin helping the receivers in a multicast group. In Section 4, we present our evaluation metrics, and simulation methodology. We present our results in Sections 5, 6, and 7. In Sections 8 and 9, we discuss the implications of our results and future work.

[1] A waypoint could be a separate machine or a program running on a shared machine.

2 STORM: Structure-Oriented Resilient Multicast

STORM belongs to the family of tree-based multicast error recovery protocols in which receivers build a hierarchical structure among themselves. Control packets (NACK, ACK, and Repair packets) follow this recovery structure rather than the multicast data delivery path.

Compared to other tree-based protocols [7, 8, 11, 17, 24], STORM has several unique characteristics. First, it was designed to support resilient multicast rather than reliable multicast. While reliable multicast protocols are designed to recover *all* data *eventually*, STORM is focused on continuous-media data such as audio and video, where a small amount of loss is tolerable and in many cases not noticeable by the user. Also, continuous-media data has an implicit deadline: if a packet arrives after the time it should have been played back, the packet is useless. In STORM, each receiver can independently set its playback point by properly sizing its buffer. The size of the buffer represents a tradeoff between reliability and interactivity. The larger a receiver's buffer, the larger the amount of time it has to recover packets and the lower its loss rate will be. However, a large buffer will also increase the playback delay, reducing interactivity. The goal of STORM is to minimize the loss rate for each receiver given its buffer size.

An additional difference between STORM and other tree-based protocols is that its recovery structure is a directed acyclic graph in which each receiver may have multiple parents (other receivers or the source). A receiver that notices a gap in packet sequence numbers sends a NACK for the missing packet up the graph toward the source. When a repair is found, it is sent down the graph toward the receivers that need it. Note that a receiver unicasts each NACK to only one of its parents at a time, not to all of its parents.

The recovery structure is built dynamically using *expanding ring search* (ERS) as receivers enter the group. ERS consists of a receiver multicasting a parent query message to the group periodically, gradually increasing the TTL of the multicast. Receivers that are already part of the structure will send replies to any parent query messages they receive. The ERS query and the reply messages contain enough information for the ERS sender to rank the repliers in terms of their likelihood of being good parents. A good parent has two qualities: First, it is able to deliver repairs in time for the child to play them back. And second, it has a low correlation of lost packets, meaning that the parent and child are less likely to lose the same packet.

Each receiver continuously reevaluates the effectiveness of its parents and changes parents if necessary to adapt to changes in group membership and/or network load. Using dynamic parent changes makes it easy to support STORM with VASPs since the dynamic binding of receivers and waypoints can be implemented using the same mechanism.

To impose a general shape on the structure and to avoid loops, each receiver has a *level*, a number that is assigned when the receiver joins the group and which remains unchanged throughout the session. Ideally, the level should be proportional to the receiver's distance from the source. In practice, we can use a receiver's hop count from the source or an estimate on the round trip time between the receiver and the source as the level. By only allowing receivers to choose parents that have lower levels, we prevent loops and

force NACKs to flow towards the source.

3 Waypoint Architecture and Protocol

3.1 Design Issues

In the previous section we reviewed STORM, a receiver-based resilient multicast protocol. In this section, we present an architecture and associated mechanisms which provide a value-added service to enhance a recovery protocol's performance.

Figure 1 presents a simple example to illustrate how waypoints can improve the performance of the STORM protocol. Without waypoints, receivers can only choose the source or other receivers as parents (Figure 1 (a)). With waypoints (Figure 1 (b)), receivers have a larger selection of machines to ask for help, and the waypoints may be in a better position than other receivers to help recover from lost packets; therefore, application performance may improve with the introduction of waypoints. To implement this architecture, we need to address two issues:

- How will end hosts invoke the service?
- How will waypoints determine when to join and leave multicast groups?

Our first concern is how end host applications can request service from the VASPs. Although we only discuss how a single VASP functions for a single application, our architecture can be extended to other value-added services and to multiple service providers. There are two main approaches that can be taken: end host applications could send requests to a centralized waypoint manager, or they could send requests to a well-known multicast address to which waypoints subscribe. A centralized waypoint manager would be both a single point of failure and a bottleneck through which all requests would have to pass.

The approach we have chosen to explore in this paper, a distributed waypoint instantiation protocol, is more robust. However, the distributed aspect raises many questions. In order to instantiate waypoints to assist a multicast group, a request message from at least one member of the group needs to be sent to a multicast address that waypoints are monitoring. Once a waypoint sees the message, it decides whether to join the group requesting help. If every waypoint were to join every group for which it received a request, the system would lack scalability. This would overload both the waypoints and the network.

In addition, having all waypoints join may not be as beneficial as having a subset of waypoints that can effectively provide repairs join. A static waypoint, i.e. a waypoint that joins the group and stays until the end of the multicast session, will waste network resources when it is not useful. This implies that there should be some method by which each waypoint can know whether or not it is helping to improve receiver loss rates once it has joined the group. If each waypoint makes independent decisions to join, they do not have control over how many other waypoints will also join. Having joined a group, a waypoint should keep track of how effective it is at providing repairs to group members. If a waypoint is not effective, it should leave the group to minimize both its use of network resources and the amount of state it needs to keep.

Waypoints use STORM to communicate with each other and with receivers. There is only one difference between standard STORM and the waypoints' version: waypoints only pick other waypoints as parents. This prevents waypoints from increasing the load on any of the receivers.

We have outlined a general service invocation mechanism consisting of a global multicast address to which service providers subscribe. End host applications will send messages to the global address to request a service. Waypoints will make independent decisions on how to respond to each message. In the following sections, we discuss the details of the waypoints' distributed management algorithms, dynamic join and leave.

3.2 Dynamic Leave Protocol

Each waypoint keeps track of the number of NACKs it has received within a time interval. In STORM, the number of NACKs received reflects how well the waypoint is doing as a parent. If a waypoint is frequently able to send repairs to answer receivers' NACKs, it will be considered a good parent and many receivers will send NACKs to it. If a waypoint receives a small number of NACKs, indicating that few receivers are using it, the waypoint drops out of the group.

3.3 Dynamic Join Protocol

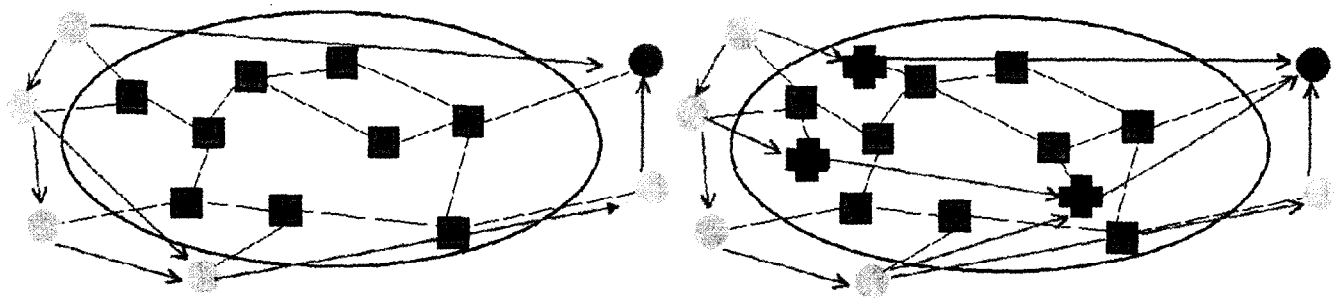
When a receiver's loss rate rises above a threshold, the receiver will multicast a request for help, called a *status message*, to a well-known multicast group which waypoints join. This multicast group acts as a rendezvous point for waypoints and receivers. After receiving a status message, a waypoint decides whether to join the multicast group specified by the receiver requesting help. There are three goals that the join protocol must meet:

1. Limit join implosion: Having a large number of waypoints join a group all at once incurs significant overhead.
2. Minimize useless joins: Prevent waypoints that will not be effective parents from joining the group.
3. Minimize oscillation: Waypoints should not endlessly join, leave, and rejoin the group.

We employ two techniques to help meet the above goals. First, receivers limit the number of waypoints that see a status message via TTL-scoping and ERS. This addresses the first two goals. Second, waypoints keep a history of their previous joins and leaves, and use that to determine their future join behavior. This addresses the second and third goal.

Receivers will employ ERS with a limited maximum TTL to send their status messages. After sending a status message, a receiver will pause its ERS to see if its loss rate improves. If the receiver's loss rate drops low enough, it will stop the ERS. If not, it will continue the ERS. The result is that only a limited number of waypoints will get a receiver's status messages. A waypoint can only enter the group if nearby receivers need help. This helps to prevent join implosion. In addition, using ERS also restricts the amount of traffic sent to the multicast group for status messages.

Further, the receiver's use of ERS guarantees that it will contact the closest waypoints, which are often the most effective, first. As in STORM, when there is a large distance between a waypoint and a receiver, it is less likely that the



(a) Without waypoints

(b) With waypoints

Figure 1: Parent recovery structure in STORM. The source is the dark grey circular node in the upper right-hand corner, receivers are light grey circular nodes, waypoints are cross-shaped, and routers are square nodes. Arrows depict the parent structure.

waypoint can provide repairs to the receiver in time to be useful. Sending status messages to nearby waypoints first increases the chances that any waypoints that join will be effective parents.

Even though it is near one or more receivers, a waypoint may not be useful. We use the amount of time a waypoint spends in a multicast group as an indication of the waypoint's usefulness. Waypoints maintain an exponentially smoothed average of how long they stay in a group. When a waypoint receives a status message, it looks at this average. If the average is above a threshold, it was previously useful and should join this time. If it has only stayed for short periods, then it was useless and should not join again. The first time a waypoint receives a status message for a group, it assumes that it will be useful and joins.

Because network conditions and the set of receivers in a group can change constantly, a waypoint could change from useful to useless or vice versa. To accommodate this, waypoints age their time averages, so they are considered less and less accurate as time passes, until they are eventually thrown out. Our current implementation does not include state aging since our simulations do not have dynamic network conditions or group membership.

Keeping track of the average amount of time spent in a group also helps to prevent join/leave oscillations. If a waypoint spends a short amount of time in the group, it is less likely to join again soon because its average time spent in the group is low. This prevents a waypoint from joining and leaving frequently.

Waypoints only need to keep a small amount of management state per multicast group: the average length of time spent in the group. Note that there is no per-receiver state. For groups in which they are currently participating, waypoints also need to maintain STORM state as well as to buffer some amount of the multicast data. The exact buffer size for a multicast group is a function of the rate at which data is being sent and the reliability requirements of receivers. For some applications, e.g. audio conferencing, data older than a few hundred milliseconds is not worth buffering. For other applications, e.g. a shared whiteboard, data ages much more slowly. The number of groups in which a single waypoint can simultaneously participate is determined in part by the mix of application buffering requirements.

4 Performance Evaluation

In this section, we discuss the design of simulation experiments to evaluate the effectiveness of our proposed schemes. We first discuss the performance indices we use for our evaluation, then describe the simulator, network model, and experimental setup. Our general goal for the evaluation is to answer the following question: Under what conditions does the proposed value-added services approach significantly improve the performance of the multicast recovery service?

4.1 Performance Indices

We evaluate waypoints in terms of their benefits to the application and the additional load they introduce to the network. The potential benefits of waypoints are twofold. First, they should lower the packet loss rate seen by applications. Further, they should lower the processing load seen by the receivers and source as the burden of error recovery is shifted to the waypoints. On the other hand, load on the network comes from waypoints joining the multicast group and from additional traffic from waypoints participating in STORM.

The metrics we use to evaluate these three areas are:

- Loss rate observed by receivers
- Change in processing load on receivers and the source
- Additional load placed on the network by waypoints

The loss rate is a direct, application-layer measure of how useful waypoints are. This is the most significant metric since if the introduction of waypoints does not improve the loss rate significantly, then there is no point in using waypoints at all. Note that in resilient multicast, a packet is considered "lost" when it either never arrives or does not arrive in time to be played back. Although we use a resilient multicast protocol in the evaluation of waypoints, our results are applicable to reliable multicast protocols. If waypoints can improve loss rates for resilient multicast, they can also reduce the time to recover packets for reliable multicast.

To measure the protocol processing load on end systems, we count the number of NACKs end systems must process. The amount of state that a receiver will keep is proportional to the number of NACKs it receives. And of course, the number of repairs a receiver will send is also determined by the number of NACKs it receives. We would hope that waypoints will take on a significant proportion of the recovery load. Likewise, the number of NACKs seen by the source will indicate the amount of protocol processing at the

source. We also expect the effectiveness of recovery to increase when waypoints are present. Recovery effectiveness is defined as the ratio of the amount of useful recovery processing to the overall amount of processing. To measure effectiveness, we will compare the number of repairs received by receivers to the total number of NACKs sent.

The main burden the waypoint places on the network is as an additional endpoint in the multicast tree. The actual increase in load is dependent on how much of a waypoint's multicast path is shared with other receivers, and is hence extremely dependent on the specifics of the topology. We estimate the additional network load caused by waypoints by counting the number of hops waypoints add to the original multicast distribution tree. Because waypoints enter and leave the group dynamically, we use ms-hops, computed by counting the number of hops in the multicast tree during each instant of the simulation. In addition, we consider the average number of waypoints present in the group as a less topology-dependent metric.

4.2 The Simulator

We use a locally written, packet-level, event-based simulator to evaluate our protocols. In the simulator, both unicast and multicast packets are routed along paths that minimize the number of hops. Each link i is characterized by two parameters: a loss rate l_i and a typical delay d_i. For each packet traversing link i, the probability that the packet gets dropped is l_i. If the packet is not dropped, it will be forwarded with a delay that is an exponentially smoothed average of values drawn from a uniform distribution between d_i and $\alpha \cdot d_i$. We fixed α at 1.2 to allow some variation in packet delay, but not enough to cause frequent packet reordering. With this model, we do not simulate delay and loss correlations among packets. Furthermore, unlike a real network, the link delay and loss properties are independent of the number of packets traversing the link. In other words, we assume that congestion is static throughout the simulation and that the multicast data stream we are simulating is not the cause of the congestion.

We used an artificial link model rather than a queue-based model to make our simulator more scalable. Using queues would not only have increased the amount of state in each router in our simulation, but also would have required us to introduce many additional flows on each link to produce the amount of congestion we wished to simulate.

Unlike most previous simulation studies on reliable multicast protocols that assume only data packet losses, in our study, control packets (NACK, Repair) are treated the same way as data packets by routers , thus, are subject to the same delay/dropping characteristics as data packets.

4.3 Network Model

We attempted to make the topologies in our simulations resemble the Internet as much as possible. Our topologies contained many campus-size networks joined together by several WAN backbones.

Backbone connectivity and delays are modeled after three actual ISPs' backbones which span the continental United States. Router connectivity information was obtained from CAIDA's Mapnet tool [13]. MCI, depicted in Figure 2(a), has 25 backbone routing sites. Sprint, in Figure 2(b), has 13 backbone routings sites. The other ISP, BBN Planet, in Figure 2(c), has 26 backbone routing sites. Overall, there are 64 backbone routing sites. Though each routing site may contain multiple routers, we modeled each site as a single logical router in our simulations. Link delays are assigned based on estimation (involving physical distance and delays known for links of similar length and capacity) and on observed round trip ping times. Backbone link delays varied from as high as 55 ms for the transcontinental links to as low as 5 ms for topologically close routers. There are three major exchange points at which all three ISPs exchange traffic. In addition, there are three other exchange points between pairs of ISPs.

We used the Georgia Tech Internetwork Topology Models (GT-ITM) [3] package to randomly generate stub domains. Intra-stub links have delays of 1 ms to 5 ms, and the delay on the link connecting the stub to the backbone is set randomly between 3 ms to 7 ms. The maximum one way delay seen between any pair of receivers is a little over 100 ms. On average, two stub domains are attached to each backbone router. Each stub domain consists of about 7 routers with a 30% chance that any pair of routers are connected. End hosts are attached to stub routers, with at most one end host per stub router.

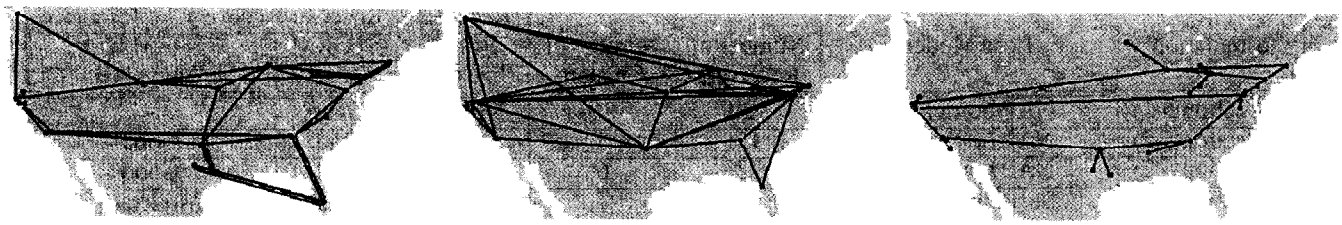
To model a variety of network conditions, we use four different loss models, presented in Table 1. In the local domain bottleneck model, most packet loss occurs inside stub networks. In the NAP bottleneck model, most packet loss occurs at transitions between stub and backbone networks. In the backbone router bottleneck model, most loss occurs in the backbones. We also used a variant of the router bottleneck model with high loss in two backbones and low loss in the backbone where the multicast source is located.

4.4 Experiments

We studied receiver performance under the above loss patterns. In this section, we present the experimental design.

Using the network topology described in the previous section, we randomly place the source in one of the stub domains. In order to determine in which circumstances waypoints are most useful in enhancing receiver performance, we varied receiver placement, both in terms of number of receivers in a multicast group, and distance between receivers. We call layouts in which there is high latency between pairs of receivers a *sparse distribution*. For example, a sparse multicast conference session might have participants located in California, Pennsylvania, Texas, and Maine. For receiver-based error recovery schemes such as STORM, receivers depend on other receivers for repairs. With a sparse receiver distribution, it is often difficult for a receiver to find another receiver capable of acting as a good parent. Under these circumstances, the latency between parent and child could be high enough such that repairs are not received by the child before its playback time. On the other hand, we call a dense layout of receivers a situation in which a receiver can usually find another receiver to act as a good parent. An example is a multicast conference session with participants in every state.

Waypoints can be placed at two general locations in the topology: directly connected to backbone routers, or connected to routers within a stub domain. Because stub domain waypoints are at the same depth as typical receivers



(a) MCI backbone (b) Sprint backbone (c) BBN Planet backbone

Figure 2: MCI, Sprint, and BBN Planet backbone connectivity.

Loss Model	Backbone-Backbone Links	Backbone-Stub Links	Stub-Stub Links
Backbone router bottleneck	High	Low	Low
Backbone router bottleneck with hot spots	High in 2 ISPs Low in 1 ISP	Low	Low
NAP Bottleneck	Low	High	Low
Local domain bottleneck	Low	Low	High

Table 1: Level of packet loss on intra-backbone links, links connecting backbones to stubs, and intra-stub links for the four loss characteristics used.

on multicast routing trees, the effect of having stub domain waypoints is very similar to having a denser distribution of receivers in a multicast group. Therefore, in our simulations, we only look at the effects of backbone waypoints.

We assume that all receivers have the same processing capabilities and buffer sizes. Waypoints have more processing power and larger buffer sizes than receivers. Unless otherwise stated, in our simulations receivers buffered data for 200 ms and waypoints buffered data for 400 ms. The source transmits at a rate of 50 packets per second to simulate audio packets with a constant interarrival time. All receivers join the multicast group at the beginning of the simulation, and stay until the end, 10 minutes later.

Two sets of simulations were run on each receiver layout and loss pattern: one set with and one set without waypoints.

5 Effectiveness of Waypoints in Reducing the Loss Rate

In this section, we discuss various factors affecting the most important evaluation index: final loss rates observed by applications after error recovery. We deduce that receiver performance can be classified into three categories based on STORM's error recovery potential and waypoints' effect on recovery. Finally, we present the final loss rates observed from simulations in each category.

The following factors contribute to receivers' error recovery performance:

- Network loss characteristics
- Network delay
- Buffer size or playback delay
- Receiver distribution

We now proceed to discuss each factor in detail. The initial loss rate is defined as the percentage of the original multicast data that did not arrive. The final loss rate, which takes repair packets into account, is the proportion of packets that did not arrive in time to be useful. If a receiver has a high initial loss rate, even when the absolute number of recovered packets is the same as a receiver with a low loss rate, its final loss rate would still be higher. Therefore, a receiver with a high initial loss rate does not see the same relative performance gain as a receiver with a low initial loss rate.

In addition, the location at which a packet is lost in the network determines which parent is capable of supplying a repair. If a packet is lost close to the source, at the top of the multicast tree, only a small subset of receivers, located on the multicast tree above the loss, would have received that packet. All other receivers would have lost it. A large number of receivers experience the same loss (correlated loss), whereas a small number of receivers are capable of supplying repairs. This increases the difficulty in recovering the packet. For example, if a packet was lost on the source's NAP to the backbone, then there is global loss. The only suitable repairer for this packet is the source, itself. If, however, a packet is lost in the lower branch of the multicast tree, more receivers would have received that packet. Therefore, more receivers are capable of sending repairs, often resulting in better error recovery.

Network delay and buffer sizes also determine error recovery performance. Because our application model has specific packet deadlines, receivers need to choose parents that are located within tolerable distances, determined by each receiver's playback delay (buffer size). If a child's playback delay is large, it can wait longer for repairs and can choose receivers that are further away as parents. The network delay between a parent and a child coupled with the child's buffer size is a fundamental limitation on parent choice. For example, if a receiver has a buffer size of 200 ms, and its round trip time to the source is over 200 ms, then it should not choose the source as a parent. If it does, it would never get a repair before playback time.

Receiver distribution can also affect error recovery performance. Sparse receiver distributions limit the number of useful parents. Again, consider a receiver with a buffer size of 200 ms, and round trip time to the source of over 200 ms. If there are no other receivers with uncorrelated loss within 200 ms round trip time, then there are no useful parents to choose from. In this case, adding a waypoint located within 200 ms of this receiver could help to reduce loss rates.

Simulation	Loss Model	Number of Receivers	Number of Clusters	Receiver Buffer Size (ms)	Receiver Recovery Performance	
					STORM	STORM with Waypoints
1 (Section 5.1)	NAP bottleneck	120	30	600	Good	-
2 (Section 5.2)	NAP bottleneck	12	3	200	Bad	Good
3 (Section 5.2)	NAP bottleneck	23	6	200	Bad	Good
4 (Section 5.3)	Backbone router bottleneck with hot spots	27	7	200	Bad	Moderate

Table 2: Loss model and receiver distribution for simulation cases.

To explore the situations in which waypoints are helpful in improving the error recovery performance of receivers, we consider the following three scenarios:

- STORM alone (i.e. recovery with receivers only but no waypoints) is already capable of providing good recovery. In this case, using waypoints does not significantly enhance receiver performance, so waypoints are not needed.
- STORM alone is not able to provide sufficient recovery. However, once waypoints are invoked, receivers see a significant reduction in loss rates.
- STORM alone cannot reduce loss rates enough. Introducing waypoints will moderately improve the receiver recovery performance.

Next, we discuss simulation results for the three scenarios. Instead of presenting simulation results for all loss models and all receiver distributions, we present a small subset of results for each case. Table 2 lists the loss model, receiver distribution, receiver buffer size, and receiver recovery performance for each simulation case to be presented. Receivers in the same stub network are referred to as a cluster.

5.1 Scenarios in which STORM by itself provides good recovery

In this section, we discuss a case in which STORM alone provides sufficient recovery. The simulation was run with the characteristics listed as Simulation 1 in Table 2. All receiver clusters are located fairly close to each other, with 30 out of 64 backbone routers having downstream receivers.

Figure 3 depicts the distribution of average initial and final loss rates seen by each cluster of receivers. The average initial and final loss rates are calculated over all receivers in the same cluster. It is reasonable to look at the average loss rate because the difference in loss rates for receivers in the same cluster was less than 1%. The horizontal axis represents the initial loss rate, and the vertical axis represents the final loss rate. Each mark on the graph represents a pair of average loss rates seen by a given cluster. The x coordinate of a mark represents a cluster's observed initial loss rates. The y coordinate of a mark represents a cluster's observed final loss rates. The diagonal line represents points at which initial loss rates are equal to final loss rates, and lost packets could not be recovered. Final loss rates are always less than or equal to initial loss rates. Therefore, all marks should fall in the area underneath the diagonal line. From Figure 3, before error recovery, receivers had loss rates in the range of 10% to 17%, with an average of

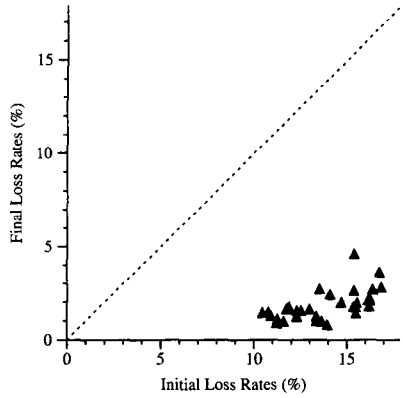


Figure 3: Distribution of initial and final loss rates of 30 clusters of receivers using STORM with no waypoints (Experiment 1 in Table 2).

13.8%. By using STORM, loss rates decreased to 0.7% to 5%, with an average of 1.84%. Receivers are able to find suitable parents because the receiver distribution is fairly dense. Also, receivers' buffer sizes are large, allowing more time for repair packets to arrive before the playback deadline. From other simulation runs, we find that even if the receiver distribution is sparse, STORM can perform well given that receivers have large buffer sizes. Further, if receivers are densely distributed, and have uncorrelated losses, recovery using STORM is also sufficient. When receivers are satisfied with recovery performance, additional help from waypoints is not necessary. In our implementation, satisfied receivers do not send status messages to waypoints to request help.

5.2 Scenarios in which waypoints help to provide good recovery

We will look at a simulation case in which receivers running STORM were not able to significantly recover from packet losses. However, given the availability of a waypoint recovery service, receivers can request for additional help. We present results from one run of STORM and one run of STORM with waypoints. Our simulation results show that the when receivers invoke waypoint recovery services, they see considerably lower final loss rates. The simulation case has the characteristics listed as Simulation 2 in Table 2. Because most packets are lost on the NAP, between clusters and their backbones, receivers within the same cluster experience the same losses (highly correlated losses). Therefore, a good parent is anyone outside the cluster. If a receiver cluster is isolated, it may not be possible to find parents located outside its cluster close enough to supply repairs by

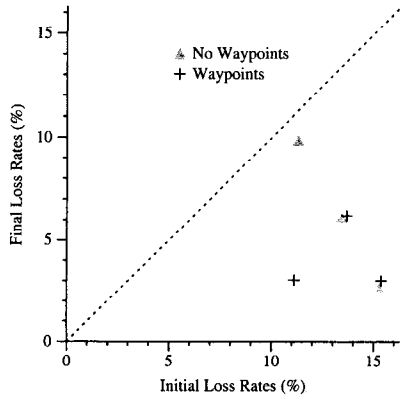


Figure 4: Distribution of initial and final loss rates, with and without waypoints for Experiment 2 in Table 2.

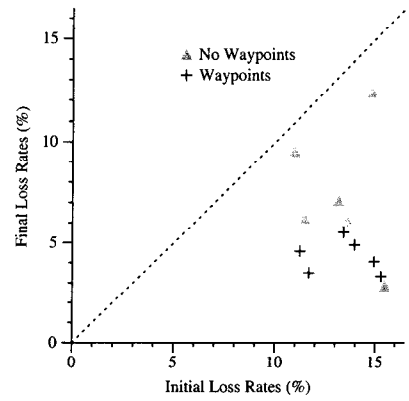


Figure 5: Distribution of initial and final loss rates, with and without waypoints for Experiment 3 in Table 2.

the playback delay deadline.

In this simulation, 3 clusters of receivers, were placed randomly in the network. There are a total of 12 receivers; 4 per each cluster. Two clusters are located fairly close to each other and to the source. The third cluster is isolated from other clusters and the source. Figure 4 depicts the distribution of the average initial and final loss rates seen by each cluster in this topology. Similar to Figure 3, each mark on the graph represents average loss rates seen by receivers in a given cluster. The x coordinates of a mark represents a receiver's observed initial loss rates. The y coordinates of a mark represents a receiver's observed final loss rates. There are two types of marks: triangle marks and cross marks. A triangle mark represents the average loss rates observed by receivers in a given cluster when using only STORM as their recovery protocol. A cross mark represents loss rates seen by a cluster of receivers using STORM and receiving assistance from waypoints. The middle cluster and the right most cluster in Figure 4 are located close to each other and the source. Receivers in the right most cluster saw initial loss rates of about 15%. Using STORM, loss rates were reduced to about 3%. However, with the presence of waypoints in the system, the final loss rates did not decrease much further. The middle cluster in Figure 4, also saw similar performance. Because these receivers are located near the source and near each other, they can leverage each other's presence for recovery. However, receivers in the left most cluster in Figure 4, saw different results. Initial loss rates were about 11%. Using STORM, loss rates only decreased to 10%. Receivers in this cluster are far away from the source and other receiver clusters enough that not all repairs arrived before playback time. When waypoints are instantiated, the final loss rates substantially dropped to about 3%.

With the addition of more receiver clusters to this topology, it is not always the case that isolated receivers will see lower final loss rates. Only when there exists suitable parents in reasonable proximity, and not sharing the same losses as receivers, can receivers see a performance improvement. Building on top of the same topology as above, we add 3 more clusters of receivers. The characteristics of this simulation is listed as Simulation 3 in Table 2. Two clusters are located within a reasonable distance to the source and to other receivers. However, the other newly added cluster is isolated from all the others. The isolated cluster from the 3 cluster topology remains isolated. The distribution of initial and final loss rates for the 6 cluster topology is depicted in Figure 5. Looking at receivers' initial loss rates and final loss rates when using STORM as the recovery protocol, most receivers see a significant decrease in final loss rates. However, the 2 isolated clusters of receivers, represented by the two top triangles in Figure 5, did not see a large decrease. In the simulation with waypoints, the final loss rates for these 2 clusters of receivers significantly decreased by 6% to 8%. Without waypoints, the 2 isolated clusters could not find suitable parents. Using waypoints as additional recovery resources, these receivers were able to improve their performance. Waypoints fill in the missing gaps when receivers cannot serve as suitable parents. In the case that multicast groups with isolated receivers are common, deploying a waypoint recovery service will significantly enhance end receivers' performance.

5.3 Scenarios in which STORM and waypoints provide limited recovery

We will now discuss a setup when neither STORM nor waypoints can provide sufficient repairs. The setup is listed as Simulation 4 in Table 2. For the backbone router with hot spots loss model, good parents are located on backbone with low losses, or on the upper branches of the multicast tree on the lossy backbones. Receivers were all located off the 2 lossy backbones. Figure 6 depicts the distribution of initial and final loss rates for each receiver. The source is located in the backbone with low losses. Initial loss rates ranged from 9% to 29%. Receivers with low initial loss rates, clusters towards the left of Figure 6, were located at the top of the multicast tree. Receivers with high initial loss rates, clusters towards the right of Figure 6, were located deeper in the multicast tree. We see that for those receivers with low initial loss rates, using only STORM gave a significant reduction in loss rates. However, for those receivers with high loss rates, STORM was able to reduce loss rates to an extent. The cluster on the far right of Figure 6 saw loss rates decrease from 29% to 19%. However, when the waypoint recovery service was invoked, final loss rates were further reduced to 14%. The combination of STORM and waypoints significantly improved receiver performance. However, the com-

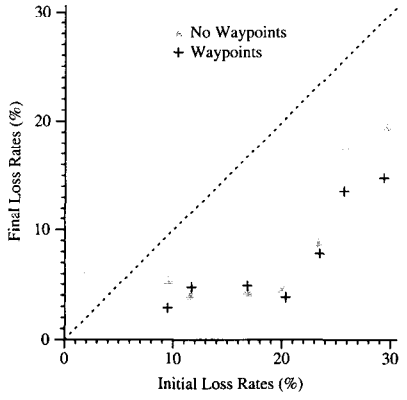


Figure 6: Distribution of initial and final loss rates, with and without waypoints for Experiment 4 in Table 2.

bination of persistently lossy backbones and limited buffer sizes constrain further improvement.

We have illustrated various performance results from using STORM and waypoints. STORM is capable of providing sufficient recovery when receivers have reasonable buffer sizes, smaller end to end delay between receivers than buffer sizes, and when there are receivers with uncorrelated losses who can be chosen as parents. However, when multicast groups have isolated clusters of receivers, a waypoint recovery service significantly enhances performance. The extent of improvement depends on the factors discussed at the beginning of this section. the number

6 NACK Rates

If waypoints are helpful, then they are chosen as parents by receivers. Because waypoints never send NACKs to receivers (only to other waypoints and the source), the more often receivers use waypoints as parents, the lower the burden from recovery traffic on receivers. Therefore, we expect the number of NACKs received by receivers to be lower when waypoints are present. Figure 7(a) shows the number of NACKs sent to receivers for the 3 cluster topology discussed in Section 5.2 and the 7 cluster topology discussed in Section 5.3. For each set of bars, the left bar is the number of NACKs sent to receivers without waypoints, and the right bar is the number of NACKs sent to receivers in the presence of waypoints. The introduction of waypoints has a profound effect: the number of NACKs sent to receivers were reduced by 33% for the 3 cluster case, and by 66% for the 7 cluster case. Note that the total number of NACKs sent should remain roughly constant as we introduce waypoints—the savings here comes from NACKs being sent to waypoints rather than other receivers.

As another measure of overhead, we also need to look at the number of NACKs the source received. Figure 7(b) shows the number of NACKs sent to the source for the 3 cluster topology discussed in Section 5.2 and the 7 cluster topology discussed in Section 5.3. For each set of bars, the left bar is the number of NACKs sent from receivers to the source without the presence of waypoints. The right bar is the number of NACKs sent to the source broken down by receivers and waypoints when waypoints are present. For the 3 cluster topology, there was not a significant difference in

Receiver Distribution	Recovery Efficiency	
	Without Waypoints	With Waypoints
3 clusters (Section 5.2)	0.37	0.55
7 clusters (Section 5.3)	0.53	0.44

Table 3: Recovery effectiveness for the 3 cluster and 7 cluster simulations with and without waypoints.

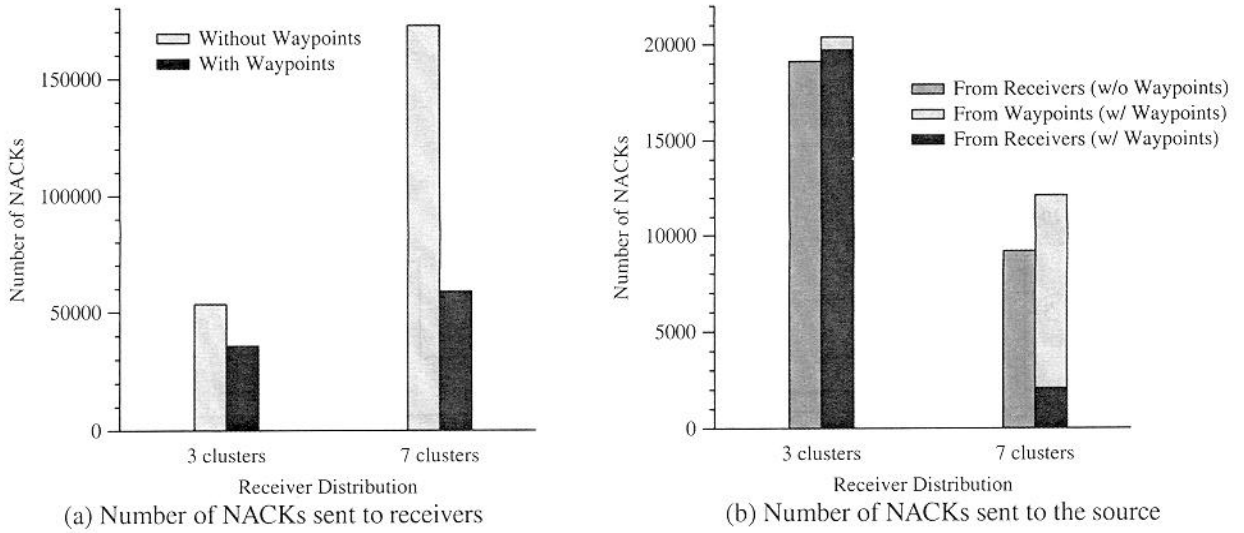
the number of NACKs sent to the source. This is because receivers close to the source continued using the source as a parent. Receivers further away from the source used waypoints as parents. However, for the 7 cluster topology, the number of NACKs from receivers to the source decreased when the waypoint recovery service was invoked. Many of the receivers found waypoints more suitable as parents than the source. But the source experienced an increase in the number of NACKs it received because the waypoints often chose the source as a parent. In addition, waypoints sent more NACKs than ordinary receivers because waypoints use larger playback buffers, meaning that they have more time to recover each packet and can send additional NACKs if a repair does not arrive in time.

It is important to note that the additional load placed on the source by waypoints is not wasted. The additional NACKs sent to the source translate into additional repairs supplied to receivers. In other words, the increase in source load translates directly into better performance for receivers. Therefore, we believe that the added load on the source is an acceptable price to pay.

A third measure of protocol overhead is whether or not waypoints increase recovery effectiveness, defined as the ratio of the number of repairs received by receivers (repairs received by waypoints are not included) to the total number of NACKs sent. Table 3 summarizes the recovery effectiveness for the 3 cluster and 7 cluster simulations. If the number of repairs received is equal to the number of NACKs sent, the effectiveness is 1. In our simulations, the effectiveness is less than 1 because NACKs and repairs can be lost and because sometimes repairs are not recovered in time. For the 3 cluster topology, recovery effectiveness increased from 0.37 to 0.55 with the addition of waypoints. The total number of NACKs sent with and without waypoints are about the same. However, the addition of waypoints increases the number of repairs received by receivers. In this scenario, waypoints enhance the effectiveness of recovery. For the 7 cluster topology, the recovery effectiveness decreased from 0.53 to 0.44 with the addition of waypoints. Although some receivers saw significant reductions in final loss rates when waypoints joined the group, the overall recovery effectiveness decreased. There was an increase in the amount of recovery traffic and recovery processing because a large number of waypoints joined the group and generated more NACKs.

7 Network Overhead From Waypoints

Next, we examine the network overhead of waypoints. We consider the overhead in terms of the increase in number of ms-hops waypoints add to the multicast tree and in terms of the increase in group size. We define ms-hops to be the num-



(a) Number of NACKs sent to receivers

(b) Number of NACKs sent to the source

Figure 7: Total number of NACKs sent to receivers and the source for the 3 cluster and 7 cluster simulations with and without waypoints.

Receiver Distribution	Overhead	
	ms-Hops	Group Size
3 clusters (Section 5.2)	2.43%	6.76%
7 clusters (Section 5.3)	17.36%	30.57%

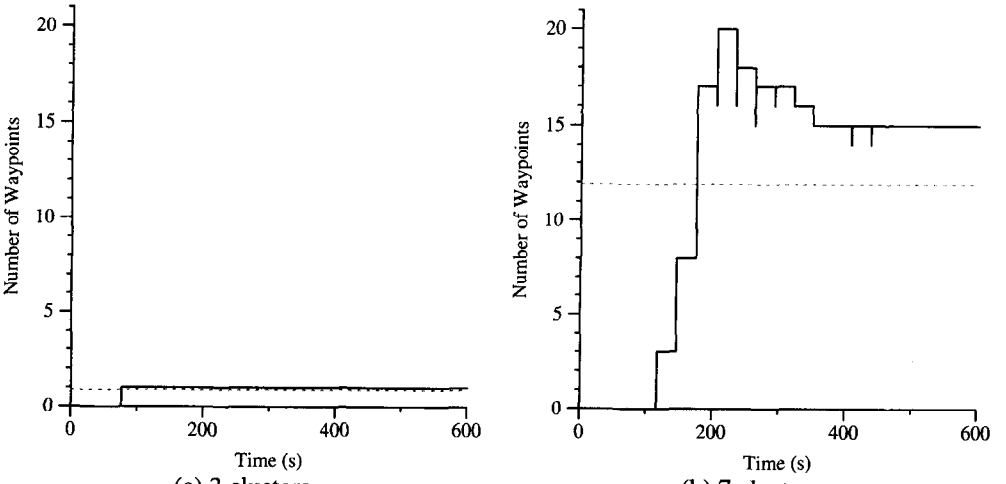
Table 4: Overhead in terms of percentage increase in the number of ms-hops on the multicast tree and percentage increase in the group size when waypoints join the multicast group for the 3 cluster and 7 cluster simulations.

ber of milliseconds a waypoint stays in the group multiplied by the number of hops that were added to the multicast tree when the waypoint joined the group. This gives us a measure of the impact of waypoints on the network load. The increase in group size is defined to be the average number of waypoints in the group divided by the total number of end hosts (receivers and waypoints) in the group. The average number of waypoints is computed by counting the number of waypoints in the group during each instant in the simulation and then dividing by the total simulation time. Looking at the increase in group size is a more topology-independent way to examine at waypoints' impact.

Table 4 lists the amount of overhead in terms of the percentage increase in the number of ms-hops and the percentage increase in the number end hosts for the simulations with 3 clusters (Section 5.2) and 7 clusters (Section 5.3). For the simulation with 3 clusters, waypoints only contributed 2.43% to the number of ms-hops in the multicast tree. However, the overhead in the number of end nodes was higher, at 6.76%. This is because most of the multicast path between the source and the waypoint was already on-tree. Adding waypoints only added a few more hops. For the simulation with 7 clusters, waypoints contributed 17.36% in ms-hops, and 30.57% in number of end nodes. Again, the increase in ms-hops was smaller than the increase in the number of end hosts since waypoints that joined the group were usually located close to the multicast tree. From our simulations, we have observed that the additional network load imposed by waypoints is usually fairly light and worth imposing in order to get the benefits of waypoints.

To explore the dynamic behavior of waypoints in more detail, we look at two time lines, Figures 8(a) and (b), depicting waypoints joining and leaving the multicast group over time. In each figure, time advances along the horizontal axis, while the number of waypoints in the group is shown by the solid line. For example, Figure 8(a) depicts the time line for the simulation with 3 clusters (Section 5.2). The first waypoint joins the group approximately 70 seconds after the beginning of the simulation, and remains in the group for the duration. After the first waypoint joins, receivers become satisfied with their performance, and do not trigger any other waypoints. In this case, the number of waypoints in the group is stable.

In some simulations, we have observed behavior similar to that shown in Figure 8(b), which depicts the time line for the simulation with 7 clusters (Section 5.3). We see that an average of 11.89 waypoints joined the group. The maximum number of waypoints in the group at once is 20. A large number of waypoints join because a larger number (compared to the simulation with 3 clusters) of receivers are unsatisfied with their loss rates. More status messages are sent, so more waypoints join the group. After the waypoint join peak is reached at 200 seconds, the number of waypoints in the multicast session oscillates between 15 and 17. The oscillation persists for another 100 seconds. Oscillation takes place either when a waypoint joins the group and leaves soon after, or else when a waypoint leaves the group but rejoins soon after. The former is caused by a status message that is sent with too large a TTL. The latter is caused by setting a waypoint's minimum NACK threshold (the smallest number of NACKs a waypoint must receive during a period of time in order to remain in the group) too high. The behavior we see in Figure S(b) suggests that parameters in the join and leave algorithm need to be tuned in order to achieve the appropriate tradeoff between performance and instability from oscillations. Another difficulty with the current algorithm is that once a waypoint leaves a group, it can immediately rejoin. Enforcing a minimum waiting period between when a



(a) 3 clusters (b) 7 clusters

Figure 8: Average (dotted line) and instantaneous (solid line) number of waypoints participating in recovery during the session.

waypoint leaves the group and when it can rejoin will help to damp the oscillations.

8 Discussion

In this paper, we propose a third-party value added approach for supporting reliable multicast. We believe this is a promising and general architecture for introducing new services and incrementally evolving the network.

Traditional network architectures usually distinguish between two types of entities: the end system (hosts) and network (switches and routers). One of the most important architecture decisions is then the division of functionalities between end systems and networks.

In the Internet architecture, the internetworking layer, or IP, implements minimal functionality — a best-effort unicast datagram service, and end systems implement all other important functionalities such as error, congestion, and flow control. Such a minimalist approach has many advantages and is probably the single most important technical reason for the Internet's growth from a small research network into a global, commercial infrastructure with heterogeneous technologies, applications, and administrative authorities. The growth of the network in turn unleashes an increasing need to develop and deploy sophisticated applications and network services (e.g. real-time, reliable multicast, anycast) which require better network support both in terms of richer functionalities and flexible QoS models. In the last several years, mechanisms and protocols have been developed to support an increasing list of functionalities at the IP layer: best-effort MxN multicast service, integrated service, mobility, security, differential service, active queue management, and explicit congestion notification.

However, an unsettling question remains: what additional functionalities should or can be added to the IP layer? The "should" part of the question is beyond the scope of this discussion as it depends on the requirements of future applications. As to the feasibility of adding more functionalities to the internetworking layer, if the current Internet architecture remains unchanged, it would be very difficult to introduce additional functionalities; and even in the case that changes can be made, it will take a long time before the new changes can be ubiquitously deployed. There are two reasons for this difficulty. First, as we are pushing the envelope of the original IP architecture, it becomes more and more difficult to reach consensus as to what should be added to the IP layer. Second, since IP is implemented at all hosts and routers, as the network becomes larger, it becomes increasingly more difficult and expensive to replace all existing hosts and routers. Realizing this difficulty and expecting more functionalities needed at the IP layer, a group of researchers are exploring a revolutionary new network architecture called active networks [21]. While active networks may provide a general solution to the problem of service deployment and network evolution, many obstacles (security, performance, state management, network stability) need to be overcome before it can become a reality.

In the foreseeable future, we will face the following the dilemma: on one hand, the internetwork layer protocol will be relatively stable and evolve slowly. On the other hand, there will be a growing need to rapidly develop and deploy sophisticated applications like Pointcast [18], reliable multicast, and caching services.

In our architecture, we introduce a type of entity called waypoints. From the network's perspective, waypoints can be either endpoints or routers. From the application's perspective, waypoints provide additional distributed computation and storage resources beyond application endpoints. Together with the communication services provided by the network, waypoints provide a value-added service to applications.

We believe such a value-added network architecture will be useful not only for introducing new services, but also for evolving networks. With the proliferation of value-added services and providers, one may find that some services are very popular, and in order to provide these services in a scalable and efficient fashion, it may be necessary to have waypoints co-located with most routers. This means the functionalities implemented by these services should be sunk into the IP level infrastructure. Since such a service is already implemented by a value-added service provider, an incremental deployment to the IP level will not interrupt the service. An interesting example is the the deployment of IP

multicast service. While it was first implemented as an overlay network called the MBone, some of the multicast routers are now *embedded* in the IP infrastructure.

In this paper, we study a simple retransmission service within the value-added network services architecture for the STORM reliable multicast protocol. It is possible to provide other interesting services with such an architecture. For example, in the context of reliable multicast, many researchers have explored the possibility of using Forward Error Correction (FEC) techniques [2, 14, 15] for error control. In FEC schemes, the amount of redundancy added to the data stream should be a function of the loss rate of the receiver and the amount of additional network bandwidth available. With heterogeneous receivers, it is difficult for the sender to decide the right balance. It is conceivable to use waypoints to add redundancy instead and tailor the amount of redundancy to a particular subset of receivers which share similar error characteristics. Another possible use of waypoints is to collect congestion information for multicast congestion control protocols.

9 Conclusion and Future Work

We have introduced the concept of third party value-added network services. Third party service providers will deploy various value-added services on machines/servers scattered all over the network, and will allow for dynamic invocation of services.

We have investigated the effectiveness of a third party value-added network service at enhancing the performance of a resilient multicast protocol, STORM. Through simulations, we have identified cases in which resilient multicast sessions using waypoints see overall lower loss rates. For sparse receiver distributions, where receivers are located far from each other, the performance gain is substantial. As the receiver distribution becomes dense, the performance gain decreases. However, independent of distribution, waypoints reduce receiver processing by shifting error recovery away from the receivers. Using dynamic join and leave protocols, waypoints can provide significant performance gains while using moderate network resources.

Future work includes exploring the parameters and design of the waypoint join and leave protocol, the effect of dynamic receivers, and the intregration of other reliable multicast protocols into the waypoint recovery service. We are also currently implementing a prototype service to run on the MBONE.

We believe that value-added network services can be efficiently provided in many network environments, with or without IP level support. These servers are key to quickly introducing new services that may not be supported by the underlying IP network. In addition, this architecture provides a seamless path for incrementally evolving the network.

References

[1] S. Bhattacharyya, J. F. Kurose, D. Towsley, and R. Nagarajan. Efficient rate-controlled bulk data transfer using multiple multicast groups. In *Proceedings of Infocom '98*, page 1172, San Francisco, California, March/April 1998.

[2] J. Bolot, H. Crepin, and A. V. Garcia. Analysis of audio packet loss in the Internet. In *Proceedings of NOSSDAV'95*, pages 163–174, April 1995.

[3] Kenneth L. Calvert, Matthew B. Doar, and Ellen W. Zegura. Modeling Internet topology. *IEEE Communications Magazine*, 35(6), June 1997.

[4] S. Y. Cheung and M. H. Ammar. Using destination set grouping to improve the performance of window-controlled multipoint connections. *Computer Communications Journal*, 19:723–736, 1996.

[5] S. Floyd, V. Jacobson, S. McCanne, C. G. Liu, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. In *Proceedings of the ACM SIGCOMM 95*, pages 342–356, Boston, MA, August 1995.

[6] H. Holbrook, S. K. Singhal, and D. R. Cheriton. Log-based receiver-reliable multicast for distributed interactive simulation. In *Proceedings of SIGCOMM'95*, pages 328–341, Boston, MA, August 1995.

[7] H. Holbrook, S. K. Singhal, and D. R. Cheriton. Log-based receiver-reliable multicast for distributed interactive simulation. In *Proceedings of SIGCOMM'95*, pages 328–341, Boston, MA, August 1995.

[8] S. K. Kasera, J. Kurose, and D. Towsley. A comparison of server-based and receiver-based local recovery approaches for scalable reliable multicast. In *Proceedings of IEEE Infocom '98*, 1998.

[9] L. Lehman, S. Garland, and D. Tennenhouse. Active reliable multicast. In *IEEE Infocom'98*, San Francisco, CA, March 1998.

[10] B. N. Levine and J. J. Garcia-Luna-Aceves. Improving Internet multicast with routing labels. In *Proceedings of ICNP '97*, 1997.

[11] B. N. Levine, D. B. Lavo, and J. J. Garcia-Luna-Aceves. The case for concurrent reliable multicasting using shared ACK trees. In *Proceedings of ACM Multimedia'96*, November 1996.

[12] D. Li and D. R. Cheriton. OTERS (on-tree efficient recovery using subcasting): A reliable multicast protocol. In *Proceedings of ICNP '98*, October 1998.

[13] Mapnet. http://www.caida.org/Tools/Mapnet/Backbones/.

[14] J. Nonnenmacher and E. W. Biersack. Reliable multicast: Where to use forward error correction. In *Proceedings of the 5th. Workshop on Protocols for High Speed Networks*, 1996.

[15] J. Nonnenmacher, M. Lacher, M. Jung, G. Carle, and E. Biersack. How bad is reliable multicast without local recovery? In *Proceedings of IEEE Infocom '98*, 1998.

[16] C. Papadapoulos, G. Parulkar, and G. Varghese. An error control scheme for large-scale multicast applications. In *Proceedings of IEEE Infocom '98*, 1998.

[17] S. Paul, K. Sabnani, and D. Kristol. Multicast Transport Protocols for High Speed Networks. In *Proceedings of Local Computer Networks*, 1994.

[18] Pointcast. http://www.pointcast.com.

[19] S. Ramakrishnan and B.N. Jain. A Negative Acknowledgment Protocol with Periodic Polling Protocol for Multicast over LANs. In *Proceedings of INFOCOM '87*, pages 502–511, 1987.

[20] T. Speakman, D. Farinacci, S. Lin, and A. Tweedly. PGM reliable transport protocol specification. Internet draft draft-speakman-pgm-spec-01.txt available at ftp://ftp.ietf.org/internet-drafts/draft-speakman-pgm-spec-01.txt, January 1998.

[21] D. L. Tennenhouse and D. J. Wetherall. Towards an active network architecture. *Computer Communication Review*, pages 5–17, April 1996.

[22] B. Whetten, S. Kaplan, and T. Montgomery. A high performance totally ordered multicast protocol, August 1994. available from research.ivv.nasa.gov as ftp at /pub/doc/RMP/RMP_dagstuhl.ps.

[23] X. Xu, A. Myers, H. Zhang, and R. Yavatkar. Resilient multicast support for continuous-media applications. In *Proceedings of the 7th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, May 1997.

[24] R. Yavatkar, J. Griffioen, and M. Sudan. A reliable dissemination protocol for interactive collaborative applications. In *Proceedings of ACM Multimedia'95*, pages 333–344, 1995.