Error Control for Continuous Media and Multipoint Applications

Christos Papadopoulos and Guru Parulkar

WUCS-95-35

December 1995

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
St. Louis MO 63130-4899

This work was supported by ARPA, the National Science Foundation, and an industrial consortium of Ascom Nexion, Bell Northern Research, NTT, Southwestern Bell, Bay Networks, Tektronix, NEC America, Samsung and NIH.

Error Control for Continuous Media and Multipoint Applications

Christos Papadopoulos
christos@dworkin.wustl.edu
(314) 935-4163

Guru Parulkar
guru@flora.wustl.edu
(314) 935-4621

Abstract:

High-bandwidth multimedia applications pose new challenges to error control. These include the support of error control for Continuous Media (CM) streams and the scalable support of error control in multipoint applications where the number of participants is large. Current error control mechanisms provide no support for the above applications. In this report we present new error control mechanisms that provide the required support.

Continuous media applications have strict timing requirements which greatly affect recovery. To support continuous media applications we have designed and implemented a point-to-point error control mechanism which features the following: (1) selective repeat retransmission, (2) conditional retransmission, (3) playout buffering to increase the time available for recovery, (4) gap-based rather than timer-based loss detection to reduce loss detection latency, and (5) data integrity information delivery to the application. We present our design, implementation and evaluation plans.

Multipoint applications with a large number of participants do not scale well because of implosion. Implosion control can be performed by the sender, the receivers, or via a hierarchy. Sender-controlled implosion is appropriate when: (a) receiver anonymity and privacy is required, (b) receivers cannot buffer and retransmit data, and (c) in one-to-many connections on connection-oriented networks where receivers are isolated from each other. In hierarchical implosion control, the receivers are structured as a virtual tree with the sender at the root. The scope of receiver messages is restricted between children and their parents. We have designed and are investigating new sender-controlled and hierarchical implosion control mechanisms. We present our designs and our plans to investigate the effectiveness and associated trade-offs of the mechanisms by using implementation and simulation in our ATM testbed.

1. INTRODUCTION

Emerging high-speed networks like ATM have stirred excitement with their potential to support multimedia, continuous media (CM) and multipoint applications. Example applications include conferencing, collaborative work, video-on-demand, information browsing, multimedia magazines, voting systems, etc. In addition to high bandwidth, future networks will provide both Quality of Service (QoS) guarantees to support CM applications and efficient multicast to support multipoint applications. However, before these new network services can be successfully realized, changes are required in the protocol layers [17], the network interface [3, 20, 21, 25, 35, 44] and the operating system [27, 31].

In this work, we focus on error control. Specifically, we examine how error control must evolve to support (a) point-to-point CM applications and (b) multipoint applications with a large number of participants. Existing error control mechanisms are unsuitable for these applications because they fail to meet two important requirements: bounded latency, required by CM applications and efficient scaling, required by multipoint applications.

Error control as implemented in existing transport protocols (e.g., TCP [39]) offers 100% reliability at the expense of latency. In CM applications the limited lifetime of the data makes this trade-off inappropriate. CM applications require error control mechanisms that will not violate the application's latency requirements. For this reason error control for CM applications (especially retransmission-based error control) has created some controversy: some researchers question the success of error control due to the limited time available for recovery, and even its necessity, due to the periodic nature of CM applications. However, we argue that retransmission-based error control can be adapted and successfully applied to CM applications with minimal cost and will also help reduce the bandwidth allocation required to maintain acceptable loss characteristics in VBR applications with high peak-to-average ratio (e.g. compressed video).

In multipoint error control, as the number of participants becomes very large, a new problem emerges, namely *implosion*. Implosion occurs when a large number of endpoints in a multipoint connection send messages simultaneously. For example several receivers may generate acknowledgments at the same time, or when a loss occurs several receivers may send retransmission requests. The impact of implosion depends on the underlying communication: in bidirectional one-to-many communication only the sender experiences implosion, whereas in many-to-many communication, all participants are affected. Implosion is detrimental to communication efficiency because of the large increase in message processing and latency, the loss of messages from buffer overflow and waste of network bandwidth. Therefore, in order to support a large number of receivers, multipoint error control must be augmented with implosion control.

In this report we investigate solutions to the above problems. Our goal is to provide insights into the issues and trade-offs as they arise in future high-speed networks with an emphasis on producing practical and efficient solutions. The contributions of this work are:

- the design and implementation of a simple, inexpensive and efficient retransmission-based error control mechanism for CM applications and the demonstration of its effectiveness
- the design, analysis and implementation of scalable multipoint error control mechanisms, augmented by a set of implosion control mechanisms that take into account the nature of the underlying network (connectionless or connection-oriented), the type of communication (one-to-many or many-to-many) and the desired performance/complexity trade-off

This report is organized as follows: Section 2 presents the perspective of our work and describes how the work fits in the design of future error control mechanisms. Section 3 presents our error control mechanism for CM applications and our ongoing evaluation. Section 4 describes the problem of implosion as it arises in error recovery and Section 5 divides implosion solutions into three approaches, namely sender-controlled, receiver controlled and hierarchical implosion control. Sections 6 and 8, present sender-controlled and hierarchical implosion control solutions respectively, and our evaluation plans. Section 7 presents a recently proposed receiver-controlled implosion solution. Finally, Section 9 concludes this report.

2. PERSPECTIVE

Multipoint multimedia applications not only have a large and diverse set of distinct requirements, but within each requirement the range of support can vary enormously. The table in Figure 1 presents some of

Application Requirement	Options				
connection configuration	one-to-one	one-to-many		many-to-many	
connection size	small (2 - 10)..... medium (10 - 100)...... large (100 or more)				
quality of service	none	loss	delay	jitter	media synchronization
error control	none	application-specified		100%	
flow control	none	variable/policy-based		fully variable	
latency	100 μSec.......... 33 mSec........ 200mSec......... 1Sec........				

Figure 1: Multipoint multimedia application requirements

these requirements. Each row in the table lists a distinct requirement of multipoint-multimedia applications and each column lists the range of support for the corresponding requirement. The shaded entries highlight the requirements of the current generation of applications. Note the large increase in both the number of requirements and the range of support for each requirement. Some requirements have a small range: the connection configuration for example, can be one of three types: point-to-point, one-to-many, or many-to-many; respective application examples are file transfer, data distribution and collaborative work. Other requirements, however, have a support range that can vary by several orders of magnitude: for example, the number of endpoints a multipoint connection can range from two to possibly thousands (for entertainment audio and video), spanning three orders of magnitude; latency requirements can vary from microseconds (RPC, PVM [43]) to milliseconds (interactive audio and video, interactive applications), or seconds (file transfer), spanning almost six orders of magnitude.

Creating a communication protocol that can support the complete range of even one such requirement is in itself a challenging task. However, multimedia applications need the simultaneous support of several such requirements. For example, interactive applications require latency guarantees *and* efficient support for multiple endpoints; PVM applications require latency guarantees, multipoint support, flow and error control; browsing applications require latency guarantees, flow control, but perhaps less than 100% error control. Some applications may even switch requirements during the lifetime of a connection: for example, when in a multimedia database search the user stops browsing and begins reliable retrieval. Supporting multiple requirements is not easy: in many cases these requirements may call for contradictory solutions: for example, to provide multipoint support a protocol may reduce per-endpoint state. However, if error and flow control are also required, the state must be increased to store sequence numbers, buffers, retransmission bitmaps, etc.

We believe that it is very difficult to design a single communication protocol to support all the above requirements efficiently. In this report we focus on the error control requirement and explore problems and solutions for error control when combined with other requirements, namely connection configuration, connection size and application latency.

3. RETRANSMISSION-BASED ERROR CONTROL FOR CONTINUOUS MEDIA APPLICATIONS

In this section we present the motivation and design of a retransmission-based error control mechanism for CM applications. First we argue that retransmission-based error control is less costly than other methods of error control and can be adapted for a wide range of continuous media applications. Then, we present the key components of our mechanism, followed by a discussion about its complexity and trade-offs. Finally, we present our ongoing evaluation of the mechanism on our ATM testbed. We conclude this section with the related work.

3.1 Motivation

With the emergence of high bandwidth-delay networks, the ability of retransmission-based error control to support continuous media applications has been questioned. The criticism is twofold: (a) retransmission is claimed to be inappropriate because it requires at least one round-trip delay (RTD) for recovery, and (b) retransmission requires CPU participation, which is claimed to be too complex for continuous media applications. The proposed alternatives are to minimize loss with (1) *peak-bandwidth allocation*, (2) *concealment*, or (3) *Forward Error Correction (FEC)* [5].

Transmitting *variable bit-rate* (VBR) traffic requiring deterministic loss and delay guarantees means that peak-bandwidth reservation is needed for the duration of the connection. For applications with high peak-to-average ratio, peak-bandwidth allocation is too expensive and inefficient. However, transmitting VBR with statistical guarantees on loss and delay while reserving bandwidth close to the average will result in a high loss rate. For example, after analyzing a compressed two-hour sci-fi action movie [29], the average bandwidth of the resulting VBR compressed video was 5.34 Mbps and the peak bandwidth almost 3 times higher (15 Mbps). These peaks were infrequent as witnessed by the coefficient of variation (0.23) and therefore reserving peak bandwidth for the duration of the connection is wasteful. On the other hand, losing data in the network would lower the quality and compromise the entertainment value of the movie. This is especially important for future HDTV.

An alternative to peak-bandwidth allocation is to reserve less than the peak bandwidth and attempt to conceal [50] lost data by taking advantage of the inherent redundancy in CM applications. Examples of concealment include substitution of lost data with previously received data, interpolation and approximation. Supporters of this approach claim that the network loss rates will be low and thus concealment will be adequate. It is clear, however, that as the loss rate increases, the effectiveness of concealment decreases. Moreover, more complex concealment methods are required as the amount of lost data increases. Therefore lowering the error rate results in better concealment results, or allows the use of less complex concealment techniques.

There are two choices for mechanisms to recover lost data: retransmission and FEC. FEC requires less time to recover, but requires greater bandwidth than retransmission because it always transmits redundant information, in contrast to retransmission which retransmits only lost data. Despite the additional latency in retransmission, we believe that with enhancements retransmission-based error control can be applied to continuous media applications, while at the same time being significantly less costly than FEC[1]. We expect that with the low loss probability in future networks, as little as a single retransmission attempt will suffice in most cases.

---

1. FEC is typically used in lossy satellite communication. We find it contradictory to opt for the costlier error control mechanism, namely FEC, when the network error rate is expected to decrease by orders of magnitude

In our research we attempt to answer the following questions:

1. Is retransmission-based error recovery possible with continuous media applications?
2. How much does recovery cost in terms of bandwidth, protocol complexity and latency?
3. Does it improve the effectiveness of concealment and by how much?

In order to answer the above questions we have designed and implemented a CM transport protocol with retransmission-based error recovery. The protocol is being evaluated in our local ATM testbed with various continuous media, including audio, un-compressed and MPEG-compressed video.

3.2 Making retransmission work

Retransmission requires the following actions to recover lost data: (1) loss detection (typically timer-based) at the sender, or receiving a retransmission request from the receiver, (2) sending a retransmission by the sender, and (3) receiving the retransmission. These actions introduce an additional round-trip delay, which may cause continuous media to miss the presentation deadline or *expire,* before a retransmission is received. In order to maximize the probability that a retransmission will arrive before data expires, the mechanism uses the following techniques: (1) playout buffering at the receiver and the sender, (2) implicit expiration of the sender retransmission buffer, (3) gap-based loss detection at the receiver rather than timer-based loss detection at the sender, and (4) conditional retransmission. In addition, the mechanism delivers data integrity information to the application to help in concealment. We describe these techniques next.

3.2.1 Playout buffering

The time available for retransmission in CM applications may be increased, with no perceptible deterioration in quality to the user by introducing limited buffering at the receiver. This is called *playout buffering*[1] and the buffering delay is called *playout delay.* In interactive applications the playout delay is limited by the perceptual tolerance of the user, which is around 200 mSec [8]. Therefore, for NTSC video a playout delay of 100 mSec allows a playout buffer of three frames as shown in Figure 2. With playout buffer-

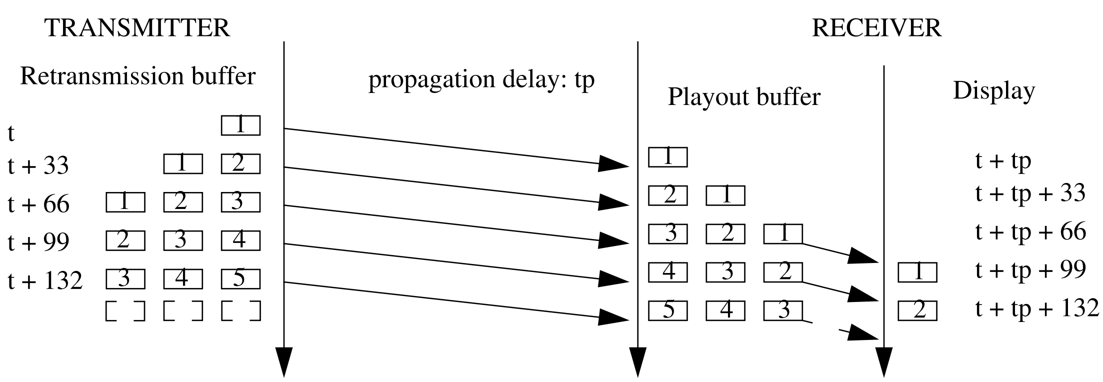


Figure 2: Buffering frames to increase time available for retransmission

ing, the time available for retransmission is increased by 99 mSec, which is sufficient for several retransmission attempts in LANs and MANs where the round-trip-delay (RTD) is about 10 - 15 mSec. It may even be sufficient for one retransmission on cross-country connections, where RTD is about 60 mSec. It is important to note, however, that the 200 mSec threshold exists in interactive applications only. In case

1. Note that the playout buffer will also act as a jitter absorption buffer

of distribution (e.g. video-on-demand) the playout buffering can be quite large (e.g. seconds) without any adverse effect on the perceived video quality. It hardly matters to a viewer if the display of a movie starts a few seconds later if a continuous, error-free playback can be guaranteed from that point on.

The size of the playout buffer is proportional to the data rate. For example, buffering 3 frames of NTSC compressed video (5 - 10 Mbps), requires a playout buffer of 0.0625 - 0.125 Mbytes. For compressed HDTV (about 20 Mbps), the size of the playout buffer for three frames is 0.25 Mbytes. These numbers are well within the capabilities of modern workstations. It remains to be seen, however, if they are within the capabilities of television set-top boxes. We are optimistic that this will not be a problem: some high-end television sets already have the capability to store frames for special effects (freeze-frame, slow motion, etc.), which indicates that such features may soon become common.

3.2.2 Implicit retransmission buffer expiration

The periodic nature of the data in CM allows the sender to estimate how long to keep data in its retransmission buffers before discarding it. Therefore, an explicit acknowledgment from the receiver is not required. The sender can simply maintain a retransmission buffer with the same number of slots as the receiver's playout buffer. Whenever new data is sent the oldest data in the retransmission buffer is discarded because it has expired. Implicit data expiration allows the sender to release its retransmission buffers when data expires without explicit indication from the receiver. It also helps avoid late retransmissions: if a retransmission request is delayed and arrives after the data has been discarded, no retransmission is sent.

3.2.3 Gap-based loss detection

Loss detection in existing protocols is typically done with time-outs. In time-out loss detection, each packet is associated with a timer at the sender. If the timer expires before an acknowledgment is received, the packet is retransmitted. The time-out values are typically large (several times the RTD), and therefore time-out loss detection is not appropriate for continuous media applications. We believe that gap-based loss detection at the receiver with an explicit retransmission request is more appropriate [30]. It is important to note, however, that gap-based loss detection is applicable only if the underlying network preserves packet sequence.

In gap-based loss detection, each packet contains a sequence number, as depicted in Figure 3. A gap is

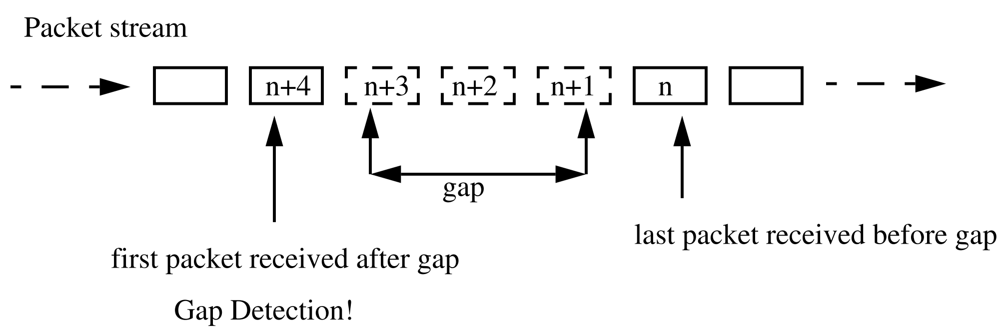


Figure 3: Gap detection at the receiver

detected when a packet arrives with a sequence number higher than expected. Gap-based loss detection has some important differences from time-out loss detection:

- a gap is detected only after another packet is received, thus gap-based loss detection is more suitable for applications that send data frequently (e.g. CM applications)
- loss detection is much faster than timer-based loss detection, if the loss is limited to a few packets
- no per-packet timers at the sender are required

Gap-based loss detection may take longer than time-out loss detection during prolonged network loss periods. However, it is questionable whether a retransmission request should be sent before the network loss period is over. With gap-based loss detection, loss is detected after the reception of a packet, so there is a good possibility that the network loss period is over (assuming no packet-discarding strategies are employed at the network [45]).

3.2.4 Conditional retransmission

Since data in continuous media has a limited lifetime, the protocol must employ a mechanism to abort retransmission when the time left before presentation is less than the RTD. This may happen after multiple retransmission attempts have failed, or if the network latency has become too large. Late retransmissions not only waste network bandwidth and CPU cycles, but contribute unnecessarily to congestion and may delay new data. To avoid late retransmissions, the mechanism must keep an estimate of the round-trip delay and the presentation time of the data at the receiver and must ensure that the presentation time is always greater than the current RTD estimate before requesting a retransmission.

3.2.5 Data integrity information delivery

The conditional, selective retransmission mechanism does not guarantee that data delivered to the application will be error-free. The mechanism, however, maintains explicit information about the integrity of the received data, which is delivered to the application. This information includes an indication to the application that data is incomplete and the location of the missing portions. We believe that such information is valuable to the application in deciding how to handle incomplete data, e.g. in applying concealment.

3.3 Complexity

The error control mechanism is a variation of selective repeat and therefore its complexity similar to the complexity of other selective repeat mechanisms [14, 16, 36]. The appropriateness of this type of error control for high-speed networks has been already demonstrated. Our mechanism includes some additional complexity, which we discuss next:

3.3.1 Retransmission decision:

The information required for the retransmission decision at the receiver are estimates of the RTD of the connection and the period of the continuous media. The RTD can be estimated by periodically bouncing a timestamped packet between the receiver and the sender. The new RTD estimate is derived after applying a smoothing function, e.g. $RTD = (a * RTD_{new}) + (1 - a) * RTD_{old}$. We expect that this simple smoothing function will suffice in connection-oriented ATM networks, where the RTD variance is expected to be low. The period of the CM stream will be one of the parameters passed to the transport protocol by the application during connection setup, in order to determine the size of the playout buffer.

3.3.2 Retransmission and playout buffer allocation and management:

The size of the retransmission and playout buffers depends on the RTD and the period of the continuous medium. The size, however, is expected to be small (see “Playout buffering” on page 5). The buffer management operations are the same as those used in other selective repeat protocols. The only difference is the increased number of buffer slots, which are managed as a FIFO queue. The state and the number of operations required to maintain a FIFO queue are small: the state consists of a head and tail pointer, and the each operation takes around 10 instructions.

3.3.3 Playout buffer status update and delivery to application

The playout buffer status consists of a bitmap indicating the presence or absence of packets. The bitmap is similar to the bitmap used in other selective repeat protocols, except that here we require one bitmap per buffer slot. In contrast to other protocols, the bitmap is made available to the application. This requires some protocol processing to prepare the *frame status*, which is a data structure containing a flag indicating loss/no loss, the loss bitmap and the packet size (assuming a constant packet size). In case where the error control mechanism is implemented in the kernel, the application interface may have to be modified to support status delivery. Ideally, the application interface must support data and status delivery in a single system call. However, this is not always possible, e.g. in the BSD socket interface.

3.4 Status and evaluation plan

We have completed the design of a CM transport protocol with the above error control mechanism and have implemented the protocol in the SunOS and NetBSD kernel, on top of IP and ATM. The protocol attempts one retransmission, which we expect to suffice in most cases. However, we plan to enhance the protocol to attempt multiple retransmissions. The user selectable parameters include the size of the retransmission and playout buffers, the packet size and the period of the continuous media stream. We are evaluating the protocol as follows:

1. We are instrumenting the protocol to measure packets lost, packets recovered and record the end-to-end latency.
2. We have implemented a tool to emulate network delay and loss. The tool consists of a machine with a kernel modified to receive, delay and/or drop and forward packets. The tool is used as shown in Figure 4 to emulate different network delays and loss models. Using this tool we are conducting

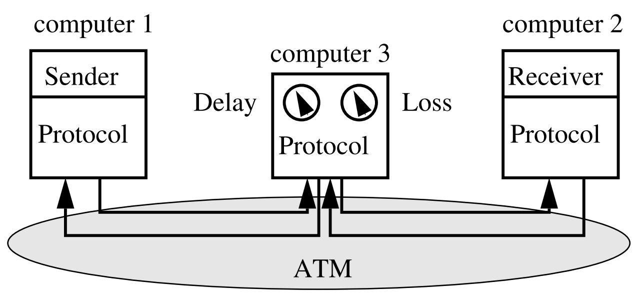


Figure 4: Protocol evaluation environment

   experiments with different network delays and loss probabilities and investigating trade-offs between the playout buffer size, RTD, loss and latency.

3. With the help of researchers at NEC in Princeton, NJ, we are assessing the effectiveness of the protocol in improving concealment with the following experiment: transmit an MPEG stream; for each received frame record the number and location of lost and recovered packets; perform off-line concealment[1] of the original MPEG stream to produce two versions, one excluding and another including data recovered by the protocol; display the two streams for subjective evaluation.

1. concealment is currently done in software and is therefore too expensive to do in real-time

3.5 Related work

In most current CM applications (especially video), no error recovery is performed [11, 12, 42]. To control losses, a mechanism is usually employed for the sender to collect feedback from receivers and adjust the transmission rate to minimize congestion and loss [7]. Other researchers propose the creation and concurrent transmission of multiple streams with different rate/quality characteristics, where receivers select and join a stream depending on the level of congestion they experience [2]. Yet other researchers propose the use of hierarchically encoded streams, where receivers receive a subset of the encoded layers [10, 41]. All the above approaches deal with flow/congestion control and are therefore orthogonal to our work. Our mechanism is an error control mechanism where we attempt to recover lost data rather than make adjustments to the data flow to avoid future congestion and loss. Therefore, we expect that our mechanism will perhaps be used in conjunction with one of the previous flow control mechanisms.

A retransmission-based error control mechanism for CM audio applications was proposed in [24]. In this work the authors have developed an end-to-end model for packet voice transmission and created a simulation to test their model. In the voice model the control time (playout buffering) between talkspurts is artificially extended when there is loss, to allow additional time for retransmission. The net effect to the audio stream is that the silence periods between talkspurts is elongated when loss occurs. Apparently, this causes no adverse audible effects for small control times. The simulation showed that retransmission-based error control is indeed feasible and the authors have shown trade-offs between loss, RTD and control time. The same model, however, may not be applicable to video because in video once playback begins the frames must be displayed at a fixed rate. Adopting a variable control-time scheme would lead to variable frame-rate video. However, we do not yet know how the end users would react to such a video stream.

The work closest to ours is Partially Reliable Streams [23]. PRS use playout buffering and conditional retransmission, but do not deliver data integrity information to the application. An implementation of PRS reportedly exists, but we unable to obtain details and evaluation studies.

In [40], the authors have developed an analytic model to study the effect of retransmission-based recovery in conjunction with concealment on ATM based networks. They show that retransmission with a 3-frame playout delay and concealment work well together even over transcontinental delays. As we mentioned earlier, we are planning to collaborate with the authors to perform experiments with MPEG data.

Other researchers have relied on FEC based error control for audio [6, 33] and video [48]. We believe that FEC may be feasible in low bandwidth streams like audio, but not for video, because of the bandwidth overhead. In addition, FEC works best if lost packets are dispersed randomly rather than being lost consecutively. However, there seems to be some controversy as to whether this is true of the Internet and the MBone: the authors in [6] find that losses appear to be random, but the authors in [47] have found losses to be correlated.

4. IMPLOSION IN MULTICAST ERROR CONTROL

In this section we first describe implosion and how it occurs in multicast error control. Then, using a simple analytic model we derive an estimate of the cost of implosion and argue that it is very high.

4.1 Motivation

Multicast networks can achieve optimal delivery of messages from a source to multiple destinations by delaying the replication of a message until it has to traverse separate paths. This results in only one message traversing any particular link in the network. The same is not true, however, for messages returned from the destinations back to the source. Ideally, such messages should be merged at the branching points as they traverse the multicast tree in the reverse direction, so that a single composite message is finally delivered to the sender. However, due to different propagation and processing delays at the receivers, it is hard to synchronize messages at the branching points. In addition, merging several messages into one is a complex operation and typically requires software processing, in contrast to replicating a single message which can be easily done in hardware. For these and other reasons (e.g. security), messages sent by the receivers are usually delivered separately to the sender.

As the number of receivers increases, so too does the number of messages returned to the sender. If the number of messages exceeds some threshold, two things may happen: (1) the sender may be swamped with messages and communication will be severely impaired or even halted, or (2) the message bandwidth at some link may exceed the available network bandwidth. This problem is called *message implosion*. An example of message implosion is depicted on the left side of Figure 5, in the context of error control. The figure shows the multicast tree for an one-to-many connection at an instant after a packet was lost on the left branch of the tree. The loss triggers retransmission requests from all receivers in the left subtree, causing implosion at the sender.

Messages that can cause implosion can be initiated by entities residing at any communication layer which sends messages back to the sender. However, we focus on implosion control in error control at the transport layer. The types of messages that can cause implosion in error control include acknowledgments, retransmission requests and redundant retransmissions. It is important to note that many of these messages may be synchronized (e.g. acknowledgments or retransmission requests) which further exacerbates the problem.

In multipoint connections, error control may cause not only implosion at the sender, but also redundancy at the receivers. This is shown at the right side of Figure 5, where some receivers receive redundant

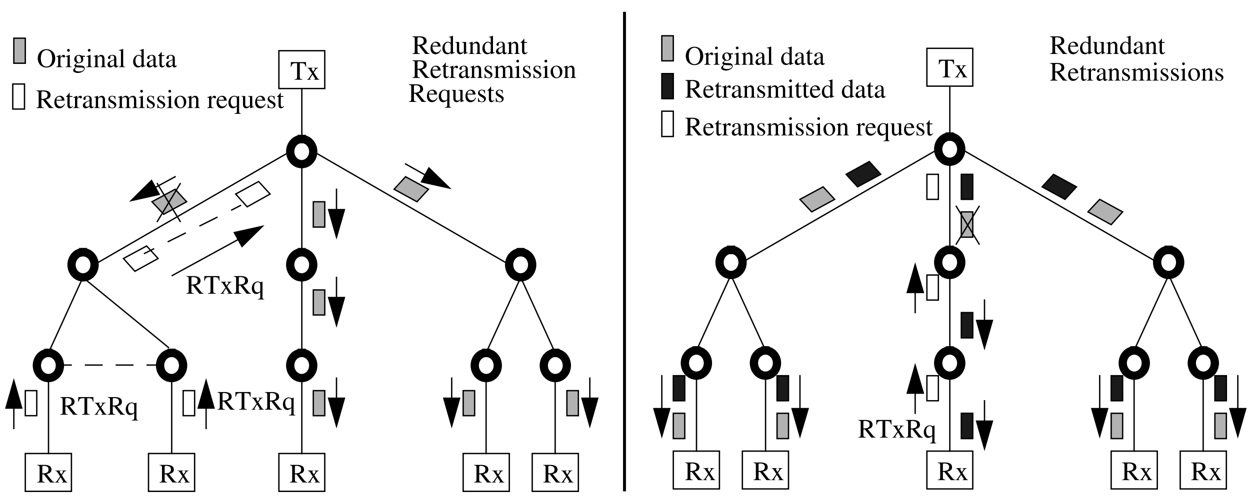


Figure 5: Redundancy in retransmission requests and retransmissions

retransmissions. Although both implosion at the sender and redundancy at the receivers are very important problems, implosion is more serious because a single endpoint can find itself being bombarded by messages from several other endpoints. Also, solving the redundancy problem requires sending retransmissions over either point-to-point connections which is wasteful, or using dynamic multicast subgroups (created to send one retransmission and then torn down) which is slow and complex.

4.2 Cost of implosion in error control

Before presenting solutions to the implosion problem, we first have to convince ourselves that the problem is worth solving. We use a simple analysis to estimate the cost of implosion in terms of messages. As a disclaimer, we point out that the analysis is not meant to accurately estimate the cost of implosion, but rather to convince us that there indeed exists a serious problem. Therefore we make assumptions that are meant to simplify the analysis, not accurately reflect real life situations. However, we believe that the analysis is accurate enough to reveal the existence of a problem, if one exists.

In our simple model, we assume a one-to-many balanced multicast tree where the endpoints lie at the leaves and all the intermediate nodes are switches. Switches receive a packet from the upstream link, replicate it and send it to all downstream links. We assume equal loss probability at the switches, which we define as the probability of a switch not receiving a packet. Therefore, a lost packet results in loss at all the receivers in the sub-tree rooted at the first switch that missed the packet. Assuming the probability a switch misses a packet is p, the height of the multicast tree (counting only the switches) is h, and each switch having the same fan-out k, then for each new packet sent the number of generated retransmission requests is:

$$n = k^h \cdot \left(1 - (1 - p)^h\right)$$

By substituting values and plotting the results, we obtain the set of graphs in Figure 6. The graphs

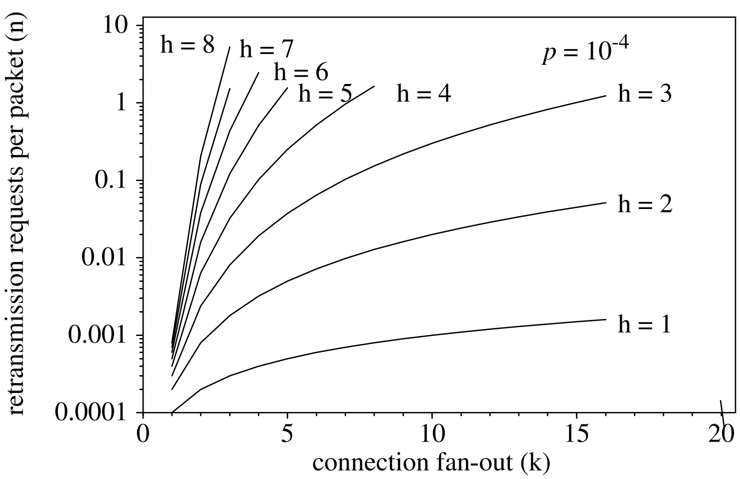


Figure 6: Retransmission requests reaching transmitter

show the number of retransmission requests received by the transmitter for every packet sent versus the fan-out at each switch. For these graphs the loss probability is set to 10^{-4}. The graphs show that the number of the received retransmission requests increase at an alarming rate as the height of the tree and the connection fan-out increase. For example, in a tree with height = 4 and fan-out = 4 (which contains in 256 receivers), the transmitter receives one retransmission request for approximately every 10 data packets it sends out. This means that in the multipoint case the sender sees an increase of 3 orders of magnitude in the number of retransmission requests compared to the point-to-point case. Such an increase will drastically reduce performance due to increased protocol processing and OS interrupts. It is important to note that the situation is very similar with flow control, or with other mechanisms generating feedback. In flow control for example, changes in the load of a link may cause all downstream receivers to send messages back to the sender creating a situation similar to packet loss.

5. CONTROLLING IMPLOSION IN MULTICAST ERROR CONTROL

In this section we begin by defining the properties of an optimal solution to implosion control and argue that implementing such an optimal solution is very hard. Thus we proceed to define a taxonomy of solutions that do not meet our optimality criteria, but are easier to implement. One of these solutions comes very close to meeting our optimality criteria.

5.1 Properties of optimal implosion control

We define an optimal solution to implosion as one that makes a multipoint connection appear to the sender as a point-to-point connection in terms of control messages and latency, regardless of the number of receivers. That is, the sender sends exactly the same number of messages and recovery takes at most the same time as in a point-to-point connection. The sender is thus presented with the illusion that it communicates with a single super-receiver, which aggregates the messages from all receivers. An optimal solution must possess certain properties in terms of messages and latency, which we define next:

Messages:

1. *Isolation*: recovery-related messages reach only endpoints that experienced loss and the retransmitting entity
2. *Redundancy*: only one retransmission request is generated from each connection subtree and only one retransmission reaches each endpoint that experienced loss

Latency:

3. *Recovery in one RTD*: a retransmission request is sent immediately by the receiver that first discovers loss and a retransmission is sent immediately by the endpoint closest to the requesting receiver

Additional conditions:

4. *Optional participation*: receiver participation in recovery is optional and recovery should be completely transparent to receivers that do not participate.
5. *Minimal message exchange*: if messages are exchanged between receivers, the number of messages is minimal and does not contribute to implosion

A significant amount of work has been done in optimizing the design and implementation of point-to-point transport protocols [18, 26]. The instruction count for the per-packet processing has been reduced to a few tens of instructions, and optimizations in memory management are resulting in only one or zero data copies of transport protocol packets. It is important that the efficiency gained in optimizing point-to-point protocols finds its way to multipoint protocols and the efficiency is not compromised by the implosion control mechanism. However, designing and implementing an optimal implosion solution is hard. State is distributed and receivers are unaware of the state or messages sent by other receivers. Clearly, receivers cannot coordinate by exchanging explicit messages, but must resort to implicit mechanisms whose parameters are hard to estimate. For example, receivers may back-off from sending messages to detect and suppress duplicates. However, estimating the right back-off delay while minimizing latency is hard, especially for receivers scattered over large geographical distances. To summarize then, the implosion problem is challenging because it requires implicit mechanisms that can accurately predict the correct actions for each receiver and minimize latency without relying on explicit message exchange.

Despite its complexity, defining an optimal solution to the implosion problem is useful for evaluating other less complex solutions. Therefore, we use the definition of the optimal solution as a metric for measuring the effectiveness of other solutions.

5.2 Network participation

Before designing implosion control mechanisms, a key question has to be answered: can the designers assume support from the network switching entities (switches, gateways, routers, etc.) for implosion control? Such support seems attractive because it allows identification and discarding of duplicates at the switching entities. However, in order to participate in implosion control the network entities must examine individual packets. This has several drawbacks:

1. Packet headers must be reassembled before they can be examined. This is very difficult at switches that switch cells (ATM).
2. Additional per-connection processing and state is required at the switching entities. These new resources must be shared and managed. In addition, the new resources will also complicate signalling protocols.
3. The network entity examining the packets must interpret the headers of the protocol, meaning that part of the protocol must reside on that particular entity. This makes software updates difficult.
4. Switches must be programmable to perform application-specified implosion control services

For the reasons listed above we believe that network participation is too complex for the current generation of switches. We therefore make the key *initial* assumption that switching entities do not participate in implosion control. As protocols and networks evolve, the possibility of network participation in implosion control must be explored. We believe that a future generation of networks will have programmable switching entities, and applications will be able to program them. For now, however, we assume that implosion control is performed by the transport protocol.

5.3 Implosion control taxonomy

A multipoint connection has two components: (1) a primary multicast channel that the sender uses to multicast data to the receivers, and (2) a secondary channel that receivers use to send control messages to the sender or other receivers[1]. We have defined a taxonomy of implosion control mechanisms based on the type of secondary channel used. The taxonomy divides the mechanisms in three categories. (1) Sender-controlled implosion, (2) Receiver-controlled implosion, and (3) Hierarchical implosion control. We introduce these next.

Sender-controlled implosion:

In sender-controlled implosion all implosion related decisions are taken by the sender. The secondary channel allows requests from receivers to go back to the sender, but not necessarily to other receivers. Typically, the sender controls implosion by: (1) providing each receiver with a back-off delay, and (2) notifying the receivers about which requests the sender has received, so that receivers can cancel their requests. Sender controlled implosion is suitable in the following cases: (1) when the sender must have absolute control of the receivers, and (2) when inter-receiver communication is not possible due to the nature of the connection (e.g. one-to-many connections in connection-oriented networks), or not allowed for security reasons (e.g. in voting systems).

Receiver-controlled implosion

In receiver-controlled implosion the secondary channel must allow messages sent by any receiver to reach all other receivers so that receivers can collaborate to control implosion. Since messages reach

1. The primary and secondary channels may sometimes be the same, as for example in many-to-many connections, where the sender and receivers use a group address for data and control.

everyone, receivers must be very careful about when to send messages. For example, on loss each receiver may calculate a back-off delay, listen for requests from others while the back-off timer is running and transmit a request only if no-one else did so. The sender may or may not be involved in the process (for example it may notify the receivers about whether it is experiencing implosion). Receiver-controlled implosion is appropriate for broadcast channels and many-to-many communication (e.g. IP Multicast, ATM many-to-many connections).

In both sender and receiver-controlled implosion, we assume that retransmissions are sent over the primary channel and reach all receivers even if they do not need them. Also we assume that in sender-controlled implosion the number of messages must be reduced below the maximum the sender can tolerate, whereas in receiver-controlled implosion the number of messages must be reduced below the maximum the sender or any receiver can tolerate.

Hierarchical implosion control

In large multicast connections a natural approach is to use a hierarchical solution. For example, the receivers can be organized into a tree hierarchy with the sender at the root. The secondary channel allows children to send messages only to their parents. Parents either combine messages from children into a single message and forward it up the tree (e.g. acknowledgments), or discard redundant messages (e.g. retransmission requests). Messages converge as they propagate up the tree and the sender finally receives a single message from each of its immediate children.

In the hierarchical approach, retransmissions can be sent by the sender to everybody over the primary channel, or by any parent to its children. The latter avoids sending retransmissions to the whole group. Parents may use the secondary channel for retransmissions if it is bidirectional, or they may insert retransmissions into the primary channel.

In the following three sections we investigate solutions based on each of the above categories. In each case we present an outline of an implosion control mechanism and point out specific trade-offs. In addition, as part of the hierarchical solution, we examine message consolidation.

6. SENDER-CONTROLLED IMPLOSION

In this section we begin by presenting a basic mechanism for sender-controlled implosion. We assume an environment where the sender is solely responsible for controlling implosion because communication among receivers is either not possible or not desirable. Following the description of the basic mechanism, we discuss trade-offs and propose a set of enhancements to the basic design. Finally, we present our evaluation plan.

Note that communication among receivers is not possible in one-to-many bidirectional connections in connection-oriented networks like ATM. We expect such connections to be frequently used in multimedia dissemination, but to the best of our knowledge, no implosion control solutions have been proposed so far for this environment.

6.1 Basic mechanism.

The key observation in sender-controlled implosion is that a retransmission request sent by a receiver remains invisible to the other receivers until the sender sends a retransmission. Therefore, to avoid implosion, each receiver has no choice but to assume that a request was sent by another receiver and delay sending a new request until it discovers otherwise. An extreme solution would be to arrange receivers in a linear chain, where each receiver waits for a retransmission initiated by the previous receiver. Such a linear chain ensures that at most one request reaches the sender at the expense of introducing very high latency (when the receiver set is large). However, typically *some* redundancy can be tolerated at the sender; in addition, not *all* receivers need to send requests at *all* times. Thus, rather than creating a linear chain, receivers can be bundled into groups where the whole group is allowed to send requests simultaneously. Such grouping makes the chain shorter and fatter, reducing latency at the expense of generating some redundant requests, which are on the order of (maximum group size). For this reason it is critical that the sender is allowed to control the size of each group to ensure that the size stays below the sender's implosion threshold.

We now present a basic mechanism based on the above observations. In this mechanism the receivers are grouped into a chain of "buckets" depending on their *RTD* from the sender, as shown in Figure 7.

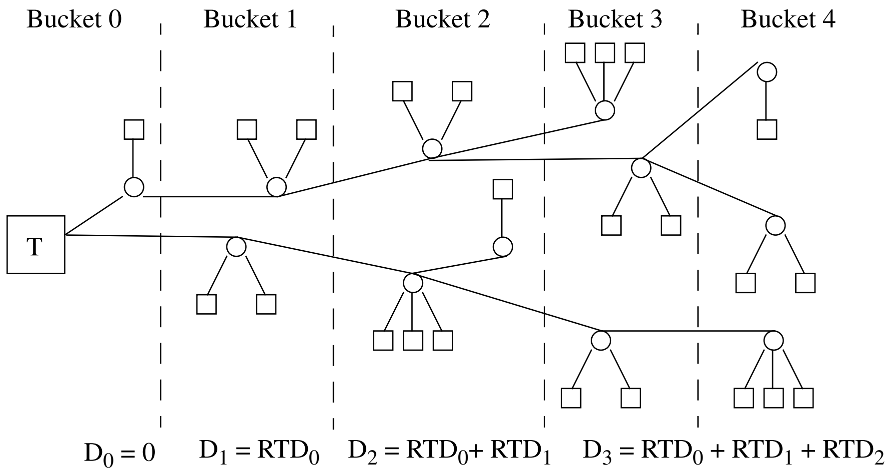


Figure 7: Grouping endpoints into buckets

Grouping receivers based on *RTD* reduces latency by preventing receivers with small *RTD* from waiting for receivers with larger *RTD*. Each bucket is associated with a value RTD_b, which is the maximum *RTD* of any receiver in the bucket and a back-off time D_b. Before sending a request, receivers in bucket b wait for a retransmission assuming that bucket *b-1* has initiated a request, and send their own requests only after a retransmission fails to arrive. D_b calculated using the following expression:

$$D_b = \sum_{k \bullet 1}^{b} RTD_{k-1}$$

We sketch the operation of the mechanism below. For brevity, we have omitted details that are not essential for understanding the operation.

Sender actions:

1. The sender maintains values for the number of buckets, the back-off timer for each bucket and a receiver histogram according to their *RTD*. When a receiver joins the connection, the sender places the receiver in the appropriate bucket based on its *RTD* and relays to the receiver the bucket *ID*, the value of the back-off timer and the value of the maximum back-off timer of any bucket (last bucket).
2. When the sender receives a retransmission request, it sends a retransmission immediately, recording the retransmission count in the request (see below). The sender ignores requests for the same data with the same or lower retransmission count for a period equal to RTD_{max}.

3. The sender keeps its retransmission buffers for an interval equal or greater than the timer value of the last bucket plus the last bucket's *RTD*. Each time a retransmission request arrives, the buffer lifetime is increased.

Receiver actions:

1. When a receiver detects a loss, it sets its back-off timer to the supplied value and waits. If a retransmission is received before the back-off timer expires, the request is cancelled.
2. If the back-off timer expires and no retransmission is received, the receiver sends a request with a retransmission *count = 1* and sets its back-off timer to the maximum back-off time (last bucket).
3. If a retransmission fails to arrive again, the retransmission count is incremented, another request is sent and the timer value is doubled. This continues for *N* iterations, after which sender or network failure is assumed.

Bucket splitting/joining:

The sender must control the size of each bucket to ensure that the bucket does not become too large. The sender splits the bucket using the following procedure:

1. The sender decides on the new values of the request delay timers for the two new buckets by dividing the receivers in two equal groups using the receiver histogram.
2. The sender multicasts the *ID* of the bucket to be split, the split threshold, the new bucket *IDs* and the new back-off values.
3. Receivers belonging to the split bucket update their state and join a new bucket. Receivers in downstream buckets add the new bucket's back-off delay to their back-off delay.

Similarly, the sender can collapse two buckets into one if it discovers that membership has dropped, by multicasting a message containing the *IDs* of the buckets to be collapsed and the new request delay timer. Collapsing buckets is highly desirable since it reduces latency and the buffering at the sender.

6.2 Limitations and trade-offs

The lack of receiver communication allows the mechanism to meet part of the isolation and part of the redundancy optimality criteria (see "5.: CONTROLLING IMPLOSION IN MULTICAST ERROR CONTROL" on page 12). The mechanism suffers from high latency, which increases in a cumulative fashion: the time-out of the last bucket is the sum of the time-outs of all previous buckets. Therefore, the mechanism penalizes receivers with high *RTD*. An important consequence of latency is that the retransmission buffers at the sender may be large. The mechanism in its present form does not scale well and the maximum number of receivers is bounded by the application latency requirements and the sender's implosion threshold. However, since the sender has full control of the receivers, the probability of implosion is minimized. The mechanism allows the sender to optimize performance by selecting a trade-off between the maximum bucket size and latency. In addition, the mechanism requires minimal implosion-related processing at the receivers thus keeping receiver complexity low.

6.3 Enhancements/optimizations

The mechanism's greatest limitation is latency, especially for receivers with high *RTD*. Next, we examine methods to reduce latency.

Latency is reduced by decreasing the total number of buckets. This can be done by increasing the size of the buckets beyond the sender's threshold, while ensuring that for every bucket the probability that more receivers than the sender's implosion threshold send requests simultaneously, is acceptably low. In order to increase the bucket size we assume that in most cases losses in the network are localized. In other words, given that a packet was lost in a switch, the probability that a subsequent packet will be lost at the same switch is high. This scenario is typical during congestion, which is expected to be the main cause of packet loss in future networks. This assumption is necessary to allow us to predict that in most cases the same set of receivers will detect the loss of a packet. For example, repeated losses at a congested switch trigger requests from the same set of downstream receivers. With this assumption, the sender can reduce the probability of implosion by placing receivers that send requests simultaneously in different buckets[1].

The preceding idea can be generalized by allowing all receivers to be distributed in arbitrary buckets to achieve more flexibility. To achieve arbitrary distribution each receiver is assigned *RTD* equal to RTD_{max}. This is necessary to allow any receiver to be placed into any bucket. The bucket back-off values become multiples of RTD_{max}, for example for $RTD_{max} = 5$ mSec, the bucket values are 0, 5, 10, 15, etc. With arbitrary receiver distribution latency is reduced by allowing larger bucket sizes, thus reducing the number of buckets. We are examining strategies for receiver placement as part of our research.

With arbitrary receiver distribution, the average latency for recovery can be made uniform for all receivers by allowing the sender to rotate the buckets periodically. For example after a bucket rotation, bucket *0* may become bucket *1*, bucket *1* may become bucket *2*, and bucket *n* may become bucket *0*. This can be accomplished by the sender periodically multicasting a bucket rotation message. Bucket rotation, however, does not reduce the maximum latency for recovery.

If receivers are distributed over large geographic distances, it may be more efficient to permanently assign receivers to buckets based on *RTD*. To avoid penalizing receivers with high *RTD*, the sender may supply a probability *p* to select receivers, which allows them to send requests immediately rather than wait for their back-off timer to expire. The sender may give higher probabilities to receivers that experience loss frequently, or may distribute the probabilities to receivers at strategic locations.

To allow receivers that experience repeated loss to send requests immediately, the sender may include the ID of the receiver to whose request is responding, with the retransmitted data. Receivers who see their IDs in the retransmitted data begin to send requests immediately and receivers who do not see their IDs follow the original back-off cycle. The process can be made adaptive, meaning that once a receiver stops seeing its ID, it slowly reverts back to the back-off mechanism.

6.4 Evaluation plan

Presently, wide-area ATM networks with the capacity for large scale connections are not available, which prevents us from evaluating the implosion control mechanisms in a real experimental environment. However, we believe we can still gain insight into the trade-offs by designing, implementing and simulating the mechanisms. Specifically, we are implementing the sender and receiver mechanisms in a transport protocol to model their behavior and use the model to simulate implosion control in a large scale connection as described next:

1. We are creating a complete specification and implementation of a protocol incorporating sender-controlled implosion on our local ATM network. The implementation will be tested using one sender and as many receivers as possible in our local environment (we expect to use between 10 and 20 receivers).

---

1. If losses in the network are random, this method may not work because there exists a probability that the number of receivers in a bucket that fire simultaneously may exceed the sender's threshold.

2. Instrument the implementation to measure the latency added by the implosion control mechanisms at the sender and the receivers. The most important measurements are: (a) the latency to send a retransmission after receiving a request, (b) the number of redundant requests and the associated processing overhead (to find the sender's implosion threshold) and (c) the processing overhead of the implosion mechanisms at the sender and receivers.
3. Create a simulation of a wide-area network using generally accepted values for bandwidth, latency and error rate, and simulate connections with a large number of receivers and various topologies (LANs, MANs, WANs). Use the measurements obtained from instrumentation to simulate the sender and the receivers.
4. Using the simulation, experiment with the various implosion control mechanisms, different network loss characteristics, and receiver topology. Obtain results for average and maximum latency, error and implosion control effectiveness, buffer requirements, and processing overhead.
5. Fine-tune and optimize the implementation based on knowledge gained from the simulation.

At the end of our evaluation we expect to be able characterize the trade-offs of the proposed implosion control mechanisms, and based on our findings, we will fine-tune the implementation of the protocol. The protocol will be documented and distributed to interested parties for experimentation.

7. RECEIVER-CONTROLLED IMPLOSION

In contrast to sender-controlled implosion, receiver-controlled implosion has drawn some attention in the last few years. One of the most significant efforts is the receiver-controlled implosion mechanisms introduced in the eXpress Transfer Protocol (XTP) in the form of *damping* and *slotting* heuristics [46]. Recently, in Scalable Reliable Multicast (SRM) [28] enhancements to the original XTP mechanisms were presented, targeted again to the Internet environment. SRM is currently gaining rapid acceptance within the IP community. We present a brief description of SRM in this section and we are planning to carry out our own evaluation of SRM in the context of ATM because in connection-oriented environments the issues and cost factors are different enough to warrant a separate evaluation. For example, the cost of setting up and maintaining many-to-many connections is drastically different in the two environments: many-to-many connections are readily supported on the Internet with IP Multicast [22], but are currently very expensive in ATM requiring the creation of several one-to-many connections (one for each endpoint).

In this section we begin by describing the operation of SRM, followed by its performance characteristics and trade-offs. We conclude with our evaluation plan.

7.1 Basic ideas

SRM assumes that receivers are capable of sending retransmission requests and retransmitting lost packets. The mechanism requires that endpoints (senders and receivers) are able to communicate via a many-to-many association, where everyone receives all requests and retransmissions. Endpoints maintain *RTD* estimates between themselves and every other endpoint in the association. The mechanism attempts to minimize latency and the number of messages by using two timers at each endpoint: a timer for sending requests and a timer for sending retransmissions. The sequence of events depicted in Figure 8 and can be intuitively described as follows: on loss, each receiver sets its request timer to a value proportional to the *RTD* between itself and the original sender. This causes receivers closer to the sender to send requests faster than other receivers. If these requests are received by other receivers before their timer expires, they cancel similar pending requests. All endpoints that can retransmit the requested data start a retransmission timer with a value proportional to the *RTD* between them and the requesting receiver. This causes the retransmission to be sent by the endpoint closest to the requesting receiver. The retransmissions also cancel similar retransmissions pending at other endpoints. The mechanism's actions are outlined next:

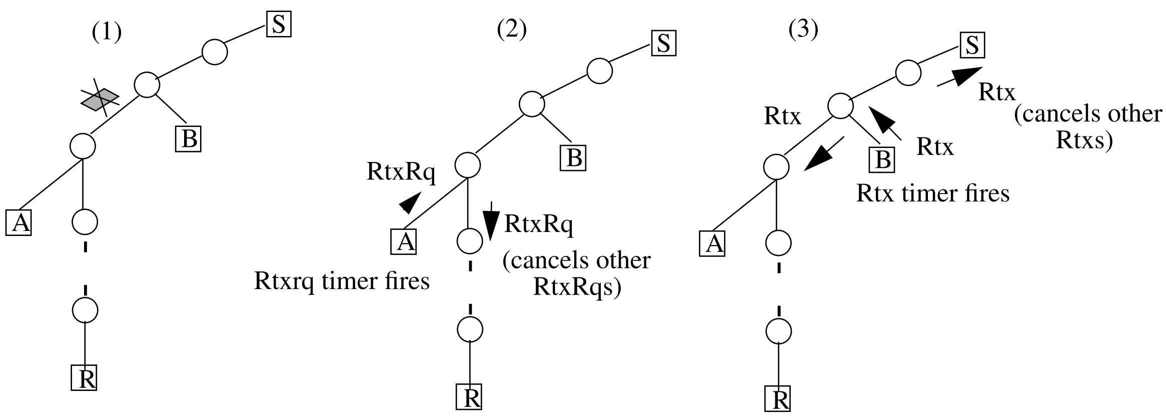


Figure 8: SRM Receiver-controlled implosion phases

Receiver actions:

1. Start with $i = 0$. If a loss is detected at receiver R, set T to a value from the uniform random interval $2^i[C_1 d_{SA}, (C_1 + C_2)\, d_{SA}]$, where d_{SA} is the propagation time from the sender S to A and C_1, C_2 are constants
2. If a retransmission for the missing packet is received during this interval, cancel the request
3. If someone else's retransmission request is received during this interval, increment i and pick a new delay interval
4. If T expires, multicast the request, double i and wait for retransmission

Sender actions:

1. Upon reception of a retransmission request, delay random time between $[D_1 d_{AB}, (D_1 + D_2)\, d_{AB}]$, where d_{AB} is the propagation delay between requesting receiver and sender and D_1, D_2 are constants
2. If a retransmission is received which was sent by someone else, cancel the repair
3. If the timer expires, multicast the retransmission

The mechanism differs from XTP in the selection of the request and retransmission back-off intervals: XTP chooses random values, while in this mechanism the intervals depend on the propagation delay of each receiver from the sender.

7.2 Performance and trade-offs

The use of a many-to-many connection as the secondary channel violates the *isolation* and the *redundancy* optimality criteria (see "5.1: Properties of optimal implosion control" on page 12). The *"recovery in one RTD"* criterion is also violated, because the mechanism introduces latency in the form of back-off delays in both sending requests and retransmissions. The extra latency can be estimated by observing that on average the first retransmission reaches the closest receiver at time $4 * p_{min}$[1] after the request is sent; if retransmission requests are dispatched immediately the latency is $2*p_{min}$. This, however, is a very good result compared to sender-controlled implosion where latency is in the order of the number of buckets.

1. $(1.5 + 0.5 + 2)*p_{min}$, where p_{min} is the propagation delay between the receiver whose request is received first, and the sender

The performance of the mechanism is closely dependent on the receiver topology and the location where a packet was lost. In the best case, a single receiver close to the sender detects loss first and sends a request which arrives in time to cancel requests from all other receivers. Thus the mechanism works best if receiver *RTDs* are dissimilar. In the worst case, however, when receiver *RTDs* are similar, the mechanism relies on widening the random interval from where the back-off timer values are selected to prevent implosion, which increases recovery latency.

The mechanism offers the following trade-offs:

- the mechanism reduces implosion by trading latency: the wider the time-out distribution the lower the chance of implosion.
- the mechanism trades absolute sender control (and therefore the guarantee that implosion will not occur) for distributed implosion processing, which offers better load balancing and scalability than sender-controlled implosion. In its present incarnation, the mechanism offers no means to the sender to request tighter implosion control.
- the mechanism trades complexity for faster recovery. Data is recovered from the closest endpoint at the expense of requiring endpoints to buffer data for retransmission, keep *RTD* estimates and execute implosion-related steps.

7.3 Evaluation plan

Our evaluation plan is similar to sender-controlled implosion: we plan to use a combination of implementation in ATM and simulation (see the previous section for details) to evaluate the mechanism's trade-offs identified above. Since many-to-many connections are more expensive in ATM than in the Internet, our investigation will also focus on characterizing the overall cost of the mechanism in terms of connection setup and maintenance.

8. HIERARCHICAL IMPLOSION CONTROL

In this section we describe a hierarchical implosion control mechanism based on structures we call *Multicast Recovery Trees* (MRTs). MRTs constitute the *secondary channel* (see "5.3: Implosion control taxonomy" on page 13) in multicast connections. For better contrast, in this section we will refer to the *primary channel* as the *Multicast Data Tree* (MDT). MRTs are created by the receivers and are composed of interconnected nodes, each containing a group of receivers. The MRT node interconnection and communication are managed by a set of MRT components and implosion is controlled by a distributed mechanism. Both the MRT node management and the implosion mechanism are discussed later. MRTs are highly scalable, offer low latency in recovery and can be used on any type of network (connection-oriented or connectionless) to provide near-optimal implosion control.

We begin by describing how receivers are structured to form MRTs. Then we describe the components required to manage MRTs, followed by a description of the implosion control mechanism running on top of MRTs. Finally we discuss the importance of consolidation and how it can be directly supported on MRTs[1]. We conclude by discussing the performance and trade-offs of MRTs followed by our evaluation plan.

8.1 MRT structure

MRTs are constructed by the receivers by forming a tree hierarchy with the sender at the root. Each node of the tree is formed by a group of receivers. Each group selects a designated receiver, called a Relay

1. Note that sender or receiver-controlled implosion mechanisms cannot directly support consolidation.

Node (RN). A RN is connected to all receivers in the group via a bidirectional one-to-many connection, and with the node's parent and children (if any), via point-to-point bidirectional connections. Receiver participation is optional and therefore the MRT topology may be different than the MDT topology. The sender may or may not be aware of the existence of MRTs. An example of a MDT and a corresponding MRT is shown in Figure 9. It is important to note that the receivers in a particular MDT are free to create more than

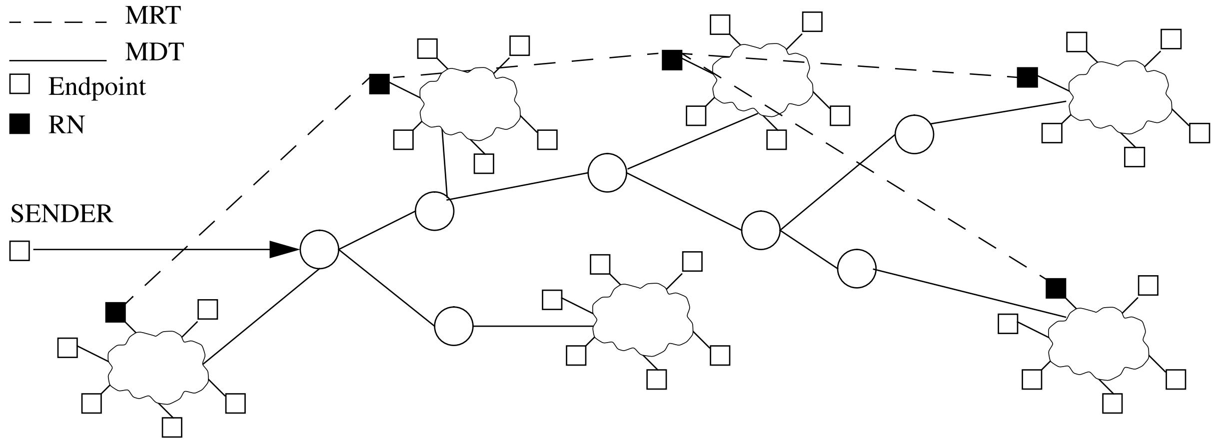


Figure 9: Multicast and recovery trees

one MRT if they need to, with different loss guarantees. For example, some receivers may be connected to a reliable MRT, while others to a best-effort MRT.

8.2 MRT components

MRTs are composed of four components. These are:

1. *MRT management* component: this component deals with the interconnection and membership operations of RNs to form the MRT. It supplies each RN with information about its parent and children. It is also responsible for reforming the MRT in case of failure.
2. *MRT communication* component: this component is responsible for communication between RNs. Typically, this is a point-to-point connection.
3. *MRT group management* component: this component deals with the formation, membership and maintenance of groups. It is responsible for notifying new receivers of the RN identity and if required, reforming the group after a failure.
4. *MRT inter-group communication* component: this component is responsible for communication between RN and its group. Typically this component is a reliable or unreliable one-to-many connection. The component is also responsible for detecting RN failure, if required.

8.3 Implosion control actions

The implosion control actions are described next. We present actions for two cases: (1) when loss occurs outside a group, and (2) when loss occurs within a group. For brevity, we omit details like how to detect endpoint failure.

Loss outside a group

1. On detecting a loss, all members in a node except RN, start a back-off timer with value greater than the largest propagation delay between any receiver in the group and RN.

2. On detecting a loss, the RN immediately sends a retransmission request to its parent RN and also multicasts the request to everyone in its group.
3. If receivers in a group receive a retransmission request from RN before their back-off timers expire, the receivers cancel their requests.
4. If the parent RN has the requested data, it relays the data to the requesting RN over the point-to-point connection. If the parent RN does not have the data, it has already sent a request to its parent. The parent RN marks that requesting child as “expecting” data.
5. When a retransmission is received from the parent, the RN relays it to the node members and to all “expecting” children.

Loss inside a group

1. If only some receivers but not RN have experienced loss, their back-off timer described earlier in step 1 will expire and one or more requests will be sent to RN. RN must have the missing data (since it did not execute step 2) and retransmits the missing data to the receivers.
2. If some receivers including RN have experienced loss, there are two options: the RN sends a request as before, which results in a transmission from receivers that have the data to the RN, or receivers stay quiet and data is recovered from the parent RN as before. In both cases the RN relays the data to the receivers, some of which will receive redundant data.

8.4 Consolidation

The actions described so far are effective in reducing redundant messages. Some types of messages, however, are not redundant and thus cannot be discarded. These messages are even more likely to cause implosion than redundant messages. Examples of messages that cannot be discarded are positive acknowledgments in a reliable protocol. An efficient method to avoid implosion from these messages is consolidation. In consolidation messages from multiple receivers are combined into a single message at intermediate nodes in the multicast tree, and the resulting message is forwarded up the tree where consolidation takes place again, until a single message is delivered to the sender. The structure of MRTs makes them highly suitable for consolidation by using the RNs as consolidating nodes. Several issues arise in message consolidation, including the following:

Selection of a message collection method:

1. Periodic: the consolidating node sets a period to wait for messages. At the end of that period, the consolidating node combines all the messages it has received and forwards the consolidated message up the tree
2. Collecting a message from each receiver: the consolidating node waits for a message from every receiver before constructing the consolidated message
3. Sender initiated: The sender initiates message collection explicitly, specifying the method of message collection the consolidating nodes should use

Consolidation operations:

A set of operations need to be defined in order to specify how a node will perform message consolidation. Some example operations are the following:

1. *MIN* $(x_1, x_2, \ldots x_n)$: may be used to consolidate fields indicating the highest packet sequence received by all receivers

2. *MAX* $(x_1, x_2, ... x_n)$: may be used to indicate the maximum window size advertized by any receiver
3. *OR* $(b_1, b_2, ... b_n)$: may be used to combine error bitmaps from all receivers to determine which packets have not been received by any receiver
4. *AND* $(b_1, b_2, ... b_n)$: may be used to combine error bitmaps to determine which packets have been received by all receivers

8.5 Performance and trade-offs

The hierarchical solution to implosion adopted by MRTs is the natural solution for connections with a large number of receivers. The hierarchy allows MRTs to offer near-optimal implosion control performance, low latency and excellent scalability. The difficulty in implementing the hierarchical solution lies in carefully specifying, implementing and integrating with the network the four components identified earlier.

We characterize the performance of MRTs as near-optimal because performance comes very close to the optimal solution we defined earlier (see “5.1: Properties of optimal implosion control” on page 12). MRTs totally eliminate implosion for losses outside the groups. MRTs deviate from the optimal solution in the following cases: (1) a retransmission originating near the root may need to traverse several hops to reach all the groups[1], (2) if loss occurs outside a group, a message from RN to the group is required to notify members that recovery is in progress, and (3) if loss occurs within the group, some receivers will receive duplicate retransmissions. Implosion may become a problem only for losses inside a group, but only if the group size is large. Typically we expect that large groups will be divided into smaller groups before implosion becomes a problem. However, if this is not feasible, large groups can control internal implosion by using one of the sender or receiver-controlled implosion mechanisms described earlier.

An important feature of MRTs is the separation of data recovery from data dissemination. The separation allows optional receiver participation in error recovery, optimized MRT topology and the decoupling of dissemination and recovery protocols. MRTs trade complexity for performance. The complexity of MRTs is higher than either sender or receiver-controlled implosion because they require additional connections and group management protocols. However, MRTs do not interfere with data transfer: for example, operations like receiver additions and deletions are mostly contained within the nodes. The RN tree is expected to be slow changing, e.g. it changes only when a new group is added or deleted, and even then, only the parent node is affected.

8.6 Evaluation plan

While the performance of MRTs appears very promising, their success depends on keeping their complexity and overhead low. Our evaluation is therefore focusing on characterizing the complexity of MRTs, by designing and implementing on our local ATM testbed the four MRT components described earlier. In designing the MRT and group management components we are examining algorithms from distributed systems. We are also investigating options and trade-offs on how RNs locate an appropriate parent and how receivers aggregate to form groups. Following the design, we are examining possible locations of these components in the protocol stack. Some options are, integrating the components with the transport protocol, the ATM signalling protocol, or implementing them as network daemons.

We are characterizing the complexity of MRTs in terms of the required connections, state, buffering and message exchange between the RNs and the receivers in a group, and the RNs with other RNs. Our evaluation is utilizing a combination of implementation and simulation similar to the one described earlier

1. This can be avoided if RNs can inject retransmissions in the MDT

in sender-controlled implosion. At the end of our evaluation we will systematically document the tradeoffs of MRTs and have efficient implementations of their components.

8.7 Related work

One of the pioneering works on reliable multicast is [13], where a centralized, reliable, totally ordered multicast protocol is described. The Reliable Multicast Protocol (RMP) [51] is an enhanced version of the protocol in [13]. Other reliable multicast protocols include [1, 9, 19, 34]. However, implosion control is not addressed in the above protocols because the number of participants is assumed to be small. Moreover, these protocols are concerned with providing 100% reliability and therefore provide no latency guarantees, making them inappropriate for many multimedia applications.

In [7] the authors describe a feedback control mechanism for variable bit rate video over the MBone. The mechanism avoids implosion by allowing the receivers to send with a period T and with some probability p a message back to the source containing QoS measure with the average packet loss they observed in that interval. However, this is a slow-reacting mechanism which aims at providing the source with some idea of the congestion experienced by the multicast group in order to adjust the output of the video coder. The mechanism is not suitable for recovery of lost data.

In [38], after studying sender initiated and receiver initiated error recovery, the authors concluded that in multicast communication the burden of error detection and retransmission request should be moved to the receivers. Three generic protocols were studied for multicast error recovery when the number of receivers is large, and it was shown that the performance of receiver initiated error control (similar to our NACK-based scheme) is substantially higher than sender-initiated error control (ACK-based) in terms of host processing requirements. However, as we have argued earlier, the problem of NACK implosion still has to be solved.

The first protocol where implosion was addressed is the eXpress Transfer Protocol (XTP) [46] with mechanisms known as the bucket algorithm, damping and slotting. SRM, which was described earlier in detail, has built upon the XTP damping and slotting mechanisms. A problem with SRM and other Internet protocols employing implosion control (e.g. TMTP, described below), is that since they require many-to-many communication, requests and retransmissions may propagate to the whole group. In an attempt to limit the scope of these messages SRM proposes and TMTP adopts the use of the TTL field. However, as we describe below, the success of this approach is highly dependent on network and connection topology.

All of the existing work has focused on receiver-controlled implosion. We are not yet aware of any published work on sender-controlled implosion.

In [4], the authors proposed a block-based acknowledgment protocol to provide reliable communication and control implosion. In this protocol, the sender knows all the receivers. The sender sends a block of packets and waits for acknowledgments from all receivers. By examining acknowledgments, the sender identifies and retransmits lost packets. The sender maintains a window of blocks and the window is advanced only when a complete block is acknowledged. Implosion, according to the authors, is reduced because receivers have to acknowledge blocks rather than individual packets. The authors have concluded that although block-based acknowledgments will reduce the ACK traffic, it will not reduce implosion without a significant increase in the interval to collect ACKs from all the receivers. In their concluding remarks the authors suggest exploring a hierarchical approach. The authors also note that the dependence of optimal transfer time on topology is very complex and present two examples to demonstrate the complexity. They do not, however, propose any solutions to reduce the complexity.

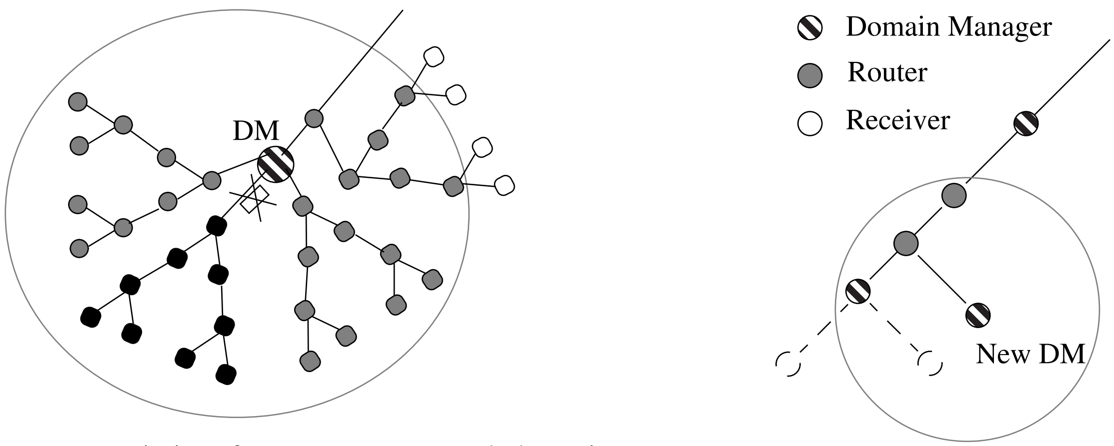
Other proposed solutions are based on a hierarchical approach similar to our hierarchical solution.

There are some important differences with our work, especially in the way the hierarchy is created and the steps taken to control implosion. We elaborate on these differences next.

In [37] a Reliable Multicast Transport Protocol (RMTP) is presented. RMTP controls implosion by dividing receivers in a static hierarchy. The source multicasts data to all receivers, but only a few designated receivers (DRs) return acknowledgments. Receivers are divided into local regions with one DR per region. Receivers periodically send their state to DRs, and DRs periodically send their state to the source. The data transmission is also periodic, and is divided into transmit time-outs. During a transmit time-out the source transmits the full current window, then stops transmission to process acks for a specified interval, then retransmits any lost data for another specified interval and finally stops again to process more ACKs. There are two main differences with our work. The first is the formation of the hierarchy: RMTP adopts a static hierarchy, whereas we propose a highly dynamic and adaptive hierarchy. Static hierarchies, however, are harder to scale and are not well-suited for applications with highly dynamic membership. RMTP has not yet proposed nor explored the issues involved with a dynamic hierarchy. As we have discussed earlier, we believe that the challenges in hierarchical solutions lie in devising dynamic, adaptive, efficient and robust methods for creating, maintaining and manipulating the hierarchical structure. The second difference of RMTP and our work is the error recovery: RMTP uses periodic state exchange to notify the sender of lost data, whereas we rely on explicit requests from the receivers.

In [32] Log-Based Receiver-reliable Multicast (LBRM) for distributed interactive simulation is presented. The protocol uses a primary logging server and a static hierarchy of secondary logging servers to provide reliability. Data is multicast from the source to all logging servers and receivers, but only the primary logging server sends acknowledgments. The receivers request lost data from their corresponding secondary logging server, and in turn the secondary logging servers request lost data from the primary logging server. In addition, in order to handle cases where data was lost in a significant number of receivers/logging servers, LBRM employs statistical acknowledgments and group size estimation to select between a single multicast retransmission or multiple unicast retransmissions. With statistical acknowledgments, the source selects a small random (and changing) set of secondary logging servers that are instructed to acknowledge each transmitted packet. Based on the number of acknowledgments received, the source chooses multicast or unicast retransmission. The main differences of LBRM and our work is that LBRM (similar to RMTP) employs a static hierarchy instead of a dynamic hierarchy and LBRM uses a distributed set of logging servers to provide reliability, which we believe to be too expensive for the majority of applications. In addition, the hierarchical structure in LBRM is such that LBRM has to select between unicast and multicast-to-all retransmissions. This creates either duplication (in case of unicast) or redundancy (in case of multicast) in most cases. In our mechanism there is no duplication and significantly less redundancy, which is limited only to the local groups, never to the whole multicast group.

In the Tree-based Multicast Transport Protocol (TMTP) [49] a dynamic hierarchical control tree is created via an expanding ring search. During the search, new domain managers (similar in functionality to our RNs) join the tree by repeatedly broadcasting a `SEARCH_FOR_PARENT` request by increasing the time-to-live (TTL) value. When one or more domain managers respond, the new domain manager selects the closest domain manager as its parent. The scope of the multicast transmissions for each domain manager is limited to within the radius of the farthest child. Similarly, the scope of NACKs from receivers is limited to the distance of its parent. Receivers multicast the limited scope NACKs after waiting for some random interval, in order to suppress duplicate NACKs. Using the TTL field to limit scope, however, faces the following problems: first, limiting multicast based on the TTL radius does not discriminate between upstream and downstream receivers and therefore, packets will travel both upstream and downstream. This causes retransmissions to be sent to both upstream and downstream receivers, even though only the downstream receivers require them, as shown in Figure 10. The second problem with the use of TTL radius lies with TMTP' s method of locating an appropriate parent: during the `SEARCH_FOR_PARENT` phase, if the clos-



Retransmissions from DM must go to dark receivers only, but reach all shaded receivers instead (SRM, TMTP)

Expanding-ring search may pick the wrong parent (TMTP)

Figure 10: Problems with using TTL to limit message scope

est domain manager willing to be a parent is located downstream the multicast tree, the expanding ring search will select it as a parent. This is obviously undesirable, because parents should be always upstream: a downstream parent domain manager will miss the same packets as a child domain manager.

9. CONCLUSIONS

In this report we have presented experimental solutions for error control for multipoint and CM applications. The solutions are, a low-cost retransmission-based error recovery method for CM applications and a set of scalable error recovery methods for multicast applications that incorporate implosion control. We have discussed in detail the mechanisms and trade-offs of these solutions and presented evaluation plans for each solution.

The contributions of this research are twofold: first, we will investigate the problems, options and the proposed solutions in detail and produce systematic evaluations of their trade-offs and performance; second, we will produce practical implementations of the chosen solutions on our ATM testbed.

REFERENCES

[1] Armstrong, S., Freier, A., Marzullo, K., “Multicast Transport Protocol,” RFC 1301, February 1992.

[2] Ammar, M., Cheung, S., Li, X., “Improving fairness in Multicast Video Distribution,” Tenth Annual IEEE Workshop on Computer Communications, September 1995.

[3] Banks, D., Prudence, M., “A High-Performance network Architecture for a PA_RISC Workstation,” IEEE JSAC, Vol. 11, No. 2, February 1993. pp. 191 - 202.

[4] Bhagwat, P., Mishra, P., Tripathi, S., “Effect of Topology on Performance of Reliable Multicast Communication,” IEEE Infocom ‘94, pp. 602-609, June 1994.

[5] Biersack, E., “Performance Evaluation of Forward Error Correction in ATM Networks,” ACM Sigcomm ‘92, vol. 22, No. 4, pp. 248-258, August 1992.

[6] Bolot, J., Crepin, H., Garcia, A., “Analysis of Audio Packet Loss in the Internet,” Proceedings of NOSDAV ‘95, pp. 163-174, April 1995.

[7] Bolot, J., Turletti, T., “A Rate Control Mechanism for Packet Video in the Internet,” IEEE Infocom ‘94, pp. 1216-1223, June 1994.

[8] Brady, P., “Effects of Transmission Delay on Conversational Behavior on Echo-free Telephone Circuits,” Bell System Technical Journal, Vol. 50, No. 1, pp.115 - 134, January 1971.

[9] Braudes, R., Zabele, S., “Requirements for Multicast Protocols,” RFC 1458, May 1993.

[10] A. Campbell, D. Hutchinson, and C. Aurrecoechea, “Dynamic QoS management for scalable video flows,” in Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), (Durham, New Hampshire), pp. 107--118, Apr. 1995.

[11] Cen, S., Pu, C., Staehli, R., Cowan, C., Walpole, J., “A distributed Real-Time MPEG Video Audio Player,” Proc. of NOSDAV ‘95, pp. 151 - 162, April 1995.

[12] Chaddha, N., Wall, G., Schmidt, B., “An End to end Software Only Scalable Video Delivery System,” Proc. of NOSDAV ‘95, pp. 139 - 150, April 1995.

[13] Chang, J., Maxemchuck, N., “Reliable Broadcast Protocols,”, ACM transactions on Computer Systems, Vol. 2, No. 3, pp.251 - 275. August 1984.

[14] Cheriton, D., “VMTP: Versatile Message Transaction Protocol,” RFC 1045, Stanford University, Feb. 1988.

[15] Tihao Chiang and Dimitris Anastassiou, “Hierarchical coding of digital television,” IEEE Communications Magazine, vol. 32, pp. 38--45, May 1994.

[16] Clark, D., Lambert, M., Zhang, L., “NETBLT: A Bulk Data Transfer Protocol,” RFC 998, MIT, 1987.

[17] Clark, D., Shenker, S., Zhang, L., “Supporting Real-Time Applications in an Integrated Packet Network: Architecture and Mechanism,” ACM SiGCOMM ‘92, Vol. 22, No. 4, pp. 14 - 26.

[18] Clark, D., Jacobson, V., Romkey, J., and Salwen, H., “An analysis of TCP processing overhead”, IEEE Communications Magazine, Vol. 27, No. 6, Jume, 1989, pp. 23-29.

[19] Crowcroft, J., Paliwoda, K., “A Multicast Transport Protocol,” Proc. ACM Sigcomm ‘88, 1988.

[20] Dalton, C., Watson, G., Banks, D., Calamvokis, C., Edwards, A., Lumley, J., “Afterburner,” IEEE Network, Vol. 7, No. 4, pp. 36 - 43, July 1993.

[21] Davie, B., “The Architecture and Implementation of a High-Speed Host Interface,” IEEE JSAC, Vol. 11, No. 2, pp. 228 - 239, February 1993.

[22] Deering, S., “Host Extensions for IP Multicasting,” RFC 1112, January 1989.

[23] Delgrossi, L., Halstrick, C., Herrtwich, R., Hoffmann, F., Sandvoss, J., Twachtmann, B., “Reliability Issues in Multimedia Transport,” Second workshop on High-Performance Communication Subsystems (HPCS’93), Williamsburg, VA, September 1993.

[24] Dempsey, B., Liebeherr, J., and Weaver, A., "On Retransmission-based Error Control for Continuous Media Traffic in Packet-switching Networks," Technical Report CS-94-09, Computer Science Department, University of Virginia, Charlottesville, Virginia, Feb. 1994.

[25] Dittia, Z., Cox, J., and Parulkar, G., "Design of the APIC: A High Performance ATM Host-Network Interface Chip," Proc. IEEE INFOCOM 95, Boston, 1995, pp. 179-187, 1995.

[26] Doeringer, W., Dykeman, D., Kaiserswerth, M., Meister, B., Rudin, H., Williamson, R., "A Survey of Light-Weight Transport Protocols for High-Speed Networks." IEEE Transactions on Communications, Nov. 1990.

[27] Druschel, P., Peterson, L., "FBUFS: A High-Bandwidth Cross-Domain Transfer Facility," Proc. of 14th Symposium on Operating Systems Principles, Asheville, NC, pp. 189 - 202, December 1993.

[28] Floyd, S., Jacobson, V., McCanne, S., Zhang, L., Liu, C., "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing," Proc. of ACM Sigcomm'95, pp. 342-356, September 1995.

[29] Garrett, M., Willinger, W., "Analysis, Modeling and Generation of Self-similar VBR Video Traffic," SIGCOMM '95, pp. 269 - 280.

[30] Gong, F., Parulkar, G., An Application-oriented Error Control Scheme for a High-Speed Transport Protocol," Technical Report WUCS-92-37, Washington University, 1992.

[31] Gopalakrishnan, R., Parulkar, G., "Efficient Quality of Service Support in Multimedia Computer Operating Systems," Technical Report WUCS-94-26, Washington University, Nov 1994.

[32] Holbrook, H., Singhal, S., Cheriton, D., "Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation," Proceedings of ACM Sigcomm '95, Vol. 25, No. 4, pp. 328-341, October 1995

[33] Jeffay, K., Stone, D., Smith, F., "Transport and Display Mechanisms for Multimedia Conferencing Across Packet-Switched Networks," Computer Networks and ISDN Systems, Volume 26, Number 10, pp. 1281-1304, July 1994.

[34] Kaashoek, M., Tanenbaum, A., Hummel, and Bal, "An Efficient Reliable Broadcast Protocol", Operating systems review, October, 1989.

[35] Kanakia, H., Cheriton, D., "The VMP Network Adapter Board (NAB): High Performance network Communication for Multiprocessors," Proc. of ACM Sigcomm '88, Vol. 18, No. 4, pp. 175 - 187.

[36] Netravali, A., Roome, W., Sabnani, K., "Design and Implementation of a High-Speed Transport Protocol," IEEE Transactions on Communications, Nov. 1990.

[37] Paul, S., Sabnani, K.,. Buskens, R., Muhammad, S., Lin, J., Bhattacharyya, S., "RMTP: A Reliable Multicast Transport Protocol for High-Speed Networks," Proceedings of the Tenth Annual IEEE Workshop on Computer Communications, September 1995.

[38] Pingali, S., Towsley, D., Kurose J., "A Comparison of Sender-initiated and Receiver-initiated Reliable Multicast Protocols", SIGMETRICS '94.

[39] Postel, J., "Transmission Control Protocol - Darpa internet Protocol Program Specification." RFC

793, September, 1981.

[40] Ramamurthy, G., Raychaudhuri, D., Performance of Packet Video with Combined Error Recovery and Concealment," IEEE INFOCOM'95, pp. 753-761, April 1995.

[41] Shacham, N., "The Design of a Heterogenous Multicast System and its Implementation over ATM," Tenth Annual IEEE Workshop on Computer Communications, September 1995.

[42] Shulzrinne, H., Casner, S., Frederick, R., Jacobson, V., "RTP: a Transport Protocol for Real-Time Applications," Internet Draft draft-ietf-avt-rtp-07, work in progress, March 1995.

[43] Sunderam, V., "PVM: A Framework for Parallel Distributed Computing," Concurrency: Practice and Experience, Vol. 2, No. 4, pp. 315-339, December 1990.

[44] Traw, C., Brendan, S., Smith, J., "Hardware/Software Organization of a High-Performance ATM Host Interface," IEEE JSAC, Vol.11, No. 2, pp.240 - 253 February 1993.

[45] Turner, J., "Managing Bandwidth in ATM Networks with Bursty Traffic," IEEE Network, pp. 50 - 58, September 1992.

[46] "XTP Protocol Definition Revision 3.6," Protocol Engines INC., 1992.

[47] Yajnik, M., Kurose, J., Towsley, D., "Packet Loss Correlation in the MBone Multicast Network: Experimental Measurements and Markov Chain Models," to appear, Infocom '96.

[48] Yavatkar, R., Manoj, L., "Optimistic Strategies for Large Scale Dissemination of Multimedia Information," Proceedings of ACM Multimedia '93, Anaheim, CA, pp. 1-8, Aug. 1993.

[49] Yavatkar, R., Griffioen, J., Sudan, M., "A Reliable Dissemination Protocol for Interactive Collaborative Applications," to appear, Multimedia '95.

[50] Masahiro Wada, "Selective recovery of video packet loss using error concealment," IEEE Journal on Selected Areas in Communications, vol. 7, pp. 807--814, June 1989.

[51] Whetten, B., Kaplan, S., Montgomery, T., "A High Performance Totally Ordered Multicast Protocol", INFOCOM '95.