

Loss Recovery at the ATM Layer for Latency-constrained Reliable Multicast

Uyen T. Nguyen and Irene Katzela
Department of Computer Science, University of Toronto
Toronto, Ontario M5S 3G4 Canada

Abstract

Many multicast applications have stringent time constraints, and require a data delivery guarantee. Currently, all existing reliable multicast protocols are at the transport layer. A cell discarded by a switch causes the loss of the entire packet, and eventually requires the whole packet to be retransmitted. Therefore, a small congestion problem could potentially escalate to a more serious one, and this in turn prolongs data recovery. The problem can be worse in the case of multicast, since many copies of the same packet are often retransmitted unnecessarily, reaching receivers which did not request the packet. This problem is called the *exposure problem*. Recovery latency would be lower if only the actual lost data were retransmitted and not the entire packet.

To reduce unnecessary data retransmission and thus recovery latency of reliable multicast applications, we propose a reliable multicast protocol in which detection of data loss and retransmission are performed at the ATM layer, instead of at the transport layer. The unit of recovery is ATM cells, instead of transport-layer packets: only missing cells are retransmitted. The proposed protocol uses the ATM physical tree structure for local recovery at the ATM layer. It is thus scalable and offers low recovery latency. In addition, retransmission operates with very minimal exposure, at the cell level, that does not aggravate existing network conditions. The built-in message stability detection scheme can be used as a rate control mechanism to help alleviate congestion, and therefore reduce recovery latency. The protocol is particularly beneficial to large-scale reliable multicast applications with stringent latency requirements such as distribution of financial data, distance learning, and Internet conferencing. We have run experiments to compare the performance of ATM-layer recovery and transport-layer recovery. Simulation results show that, with ATM-layer recovery, the average retransmission delay

is significantly lower, and the connection throughput¹ is higher under congestion.

1 Introduction

Several multicast applications require both data delivery guarantee and stringent time constraints. Examples of such applications include distribution of stock quotes and other financial data, distance learning, and Internet conferencing. To support these applications, we need a reliable multicast protocol that performs data recovery, i.e. detection and retransmission of loss data, with low latency.

In ATM networks, possible sources of data loss are (1) errors on the transmission media; (2) processing errors in switching nodes and end points; and (3) discarding cells for congestion control. Given the current technology, the physical media implementing ATM, switches and terminals have very low error rates. Cell loss due to the first two sources is thus almost insignificant. The major source of cell loss is cell discarding for congestion control.

Currently, the ATM layer does not provide any mechanisms for retransmission. Reliable delivery services are implemented as a layer on top of the ATM layer. As a result, a cell discarded by a switch causes the entire packet to be retransmitted. Sources must retransmit the discarded packets in addition to the new packets, and this only exacerbates the situation, especially when the network is heavily congested. To alleviate the problem, drop policies such as Early Packet Discard or Partial Packet Discard can be used to discard cells on a packet basis. However, one lost cell still requires the retransmission of the entire packet².

¹ We define *connection throughput* P as $P = S/T$, where S is the size of the data file to be transmitted by the source, and T is the time for the file to be received by *all* the receivers of the multicast group.

² In a TCP/IP-over-ATM network, large packets (13 cells long) generate 53% of all cells sent on the connection [1].

In the case of multicast, packets are often retransmitted unnecessarily, i.e., the exposure problem, worsening the congestion situation and delaying loss recovery.

Many transport-layer reliable multicast protocols have been proposed, and their unit of recovery is packets. The recovery would be faster if only the actual lost data were retransmitted instead of the whole packet.

We thus propose a reliable multicast protocol called REMAT (REliable Multicast at the ATm layer) in which detection of data loss and retransmission are performed at the ATM layer, instead of at the transport layer. The unit of recovery is ATM cells, instead of transport-layer packets. The new protocol bypasses packet discard mechanisms, and retransmits only missing cells. It uses the tree structure of the ATM physical connection for local recovery at the ATM level. A receiver³ requests missing cells from its local server in the multicast tree, an ATM switch. As a result, the protocol is scalable, and has low recovery latency, as will be shown in this paper. Moreover, REMAT has a built-in feedback control mechanism for buffer management at switches. This feedback control mechanism can be used to inform the source of receivers' reception rates⁴. The source can then adjust its transmission rate, to relieve congestion and speed up recovery. All the above operational features of REMAT enable it to support effectively reliable multicast applications with stringent latency requirements.

We designed and carried out experiments to compare the performance of ATM-layer recovery and transport-layer recovery in terms of retransmission delays and connection throughputs. Simulation results indicate that, with ATM-layer recovery, the average retransmission delay is significantly lower. ATM-layer recovery performs better since only lost cells are retransmitted, instead of whole packets. Also, request packets and retransmitted packets of transport-layer recovery incur overheads from packet segmentation and reassembly between the transport layer and the ATM layer. Under congestion, ATM-layer recovery also offers better connection throughputs.

³ In REMAT protocol, the term "receiver" denotes terminal equipment connected to application users receiving data. The terminal equipment is responsible for packet segmentation and reassembly. ATM cells are transparent to application users, but visible to terminal equipment. It is thus in charge of cell loss detection.

⁴ This is very useful for services with no explicit rate control such as VBR or UBR.

In this paper, we present a basic version of the REMAT protocol, and show that ATM-layer recovery can provide latency-constrained multicast applications with lower retransmission delays than transport-layer recovery. Some issues are still open for further studies, such as determining the protocol parameters for optimal performance, and handling lost request/repair cells efficiently. We are in the process of fine-tuning and adding enhancements to the protocol.

The paper is organized as follows. Section 2 provides a literature survey on reliable multicast. In section 3, we describe our new protocol for reliable multicast at the ATM layer, and discuss its implementation issues. Results of the experiments that compare the performance of ATM-layer recovery and transport-layer recovery are presented and analyzed in section 4. Section 5 concludes the paper, as well as outlines open issues and future planned enhancements of the protocol.

2 Related Work

A large number of reliable multicast protocols have been proposed in the literature. All these protocols work on top of the network multicast service, at the transport layer, to provide delivery guarantees. They can be broadly classified into four different categories: sender-based, ring-based, group-based, and tree-based. This classification is based on the agent(s) performing data retransmission, and on the structure of the recovery group.

2.1 Sender-based Protocols

Depending on the protocol, receivers send acknowledgments (ACKs) for successfully received packets, or negative acknowledgments (NACKs) for lost packets, to the sender. The sender subsequently retransmits missing packets. This approach is not scalable: the sender assumes all retransmissions and the recovery latency is thus high. Examples of sender-based protocols are XTP [2], and SCE [3].

2.2 Ring-based Protocols

The receivers are arranged in a logical ring topology. One of the receivers is selected to be a central representative. The central representative is responsible for sending ACKs back to the sender, and retransmits lost packets when requested by other receivers. The responsibility of the

central representative rotates among the receivers in order to provide fault tolerance. This approach is suitable only for small multicast groups: the central representative can be a bottleneck. It is also a failure-prone point. RMP [4], and RBP [5] are examples of protocols in this category.

2.3 Group-based Protocols

A member which has missed a packet multicasts a NACK to the entire group. Any group member that has the requested data can multicast the data. There needs to be a suppression mechanism to minimize *request implosion* (several receivers request the same data), and *repair implosion* (several members retransmit the same data). The suppression mechanism can be timer-based (as in SRM [6]), or probabilistic (as in TRM [7]). Exposure is also a potential problem since retransmission is done via multicast. Group-based protocols use global recovery, and thus do not scale to large multicast groups.

2.4 Tree-based Protocols

These protocols use the tree structure for local recovery. The receivers are organized into local regions, each with a representative. This representative is called a Designated Receiver (DR), or Domain Manager (DM). The local regions are arranged in a hierarchical tree structure.

The receivers in a local region request lost packets from their respective DR. The DR retransmits the missing packets to the receivers in the local region. If the DR does not have the requested packets, it sends a request to the next-level DR, and so on, until it reaches the sender.

The tree-based approach solves the request/repair implosion problem by design; and reduces end-to-end latency using local recovery. This approach thus scales well for large multicast groups. Example tree-based protocols are RMTP [8], TMTP [9], LBRM [10], and LGMP [11].

The tree-based approach, however, requires the DRs to do subtree multicast. This is not possible with the current IP multicast routing. To enhance IP multicast with the subtree multicast capability, several protocols have been proposed that require modifications to the IP routers. The modifications allow the routers to guide requests and repair packets to the intended destinations. Typical examples of router-assisted protocols are LSM [12], Search Party [13], and AIM [14].

3 Loss Recovery at the ATM Layer: REMAT Protocol

Our reliable multicast protocol, REMAT, also uses local recovery based on the tree structure. However, it goes in another direction: it is the first to suggest loss recovery at the ATM layer. The unit of recovery is ATM cells, instead of transport-layer packets, and only missing cells are retransmitted. The protocol uses the tree structure of the ATM connection, the routing tree, for local recovery. Switches are selected to be Retransmission Servers (RS) that are arranged hierarchically based on the routing tree. Receivers in a subtree detect cells lost and send requests to their respective RS, and the RS retransmits the requested cells. This tree-based approach minimizes request implosion and exposure, and eliminates repair implosion. As a result, REMAT has low recovery latency, and scales well for very large groups whose members are distributed in geographically diverse locations.

Note that reliable multicast connections using REMAT should bypass packet discard mechanisms such as Early Packet Discard or Partial Packet Discard at ATM switches. There is no need to throw away the rest of the packet when only a few cells in it are lost, since these cells will be retransmitted to complete the packet.

In designing the protocol, we make the following assumptions:

- The routing tree is static, because ATM requires a virtual circuit to be established before data transmission.
- There exists a multicast routing algorithm such as PIM [15], or CBT [16] that constructs the routing tree. This is outside the scope of our reliable multicast scheme.
- All receivers have similar processing capacity. This assumption is only for performance issues. The protocol still works correctly when receivers have different processing capabilities. However, the processing rate might be slowed down to the speed of the slowest receiver. In this case, one solution is to divide the original multicast group into smaller groups, each group consisting of receivers having similar processing speed.

We have addressed the following issues in the design of the new reliable multicast protocol:

1. *Adding sequence numbers to ATM cells of reliable multicast connections.* ATM cells do not contain sequence numbers that are necessary for receivers to detect lost cells for the purpose of cell retransmission⁵. Therefore, it is needed to add sequence numbers to ATM cells of reliable multicast connections.
2. *Request and retransmission algorithm.* The algorithm should use local recovery to be scalable: low recovery latency with minimal implosion and exposure.
3. *Message stability detection.* Each RS is given a cache of a configurable and *limited* size to buffer data of the multicast connection for retransmission. There must be a mechanism allowing RSs to free the buffer space of data no longer needed for retransmission.
4. *Identifying the new types of data and control cells of reliable multicast connections.* Data cells of reliable multicast applications contain sequence numbers, and require special processing at member nodes. Therefore, reliable multicast data cells need to be tagged differently from data cells of unicast and unreliable multicast connections. In addition, some new types of control cell are created for use by the new reliable multicast protocol, and they need to be identified as well.

These issues will be discussed in detail below.

3.1 Adding Sequence Numbers to ATM Cells

ATM cells do not contain sequence numbers that are needed for receivers to detect lost cells for cell retransmission. Therefore, we propose to use the first two bytes of the ATM cell payload to store sequence numbers. For example, if AAL5 were used, then the new ATM cell format would consist of a 5-byte header as usual, a 2-byte sequence number, and a 46-byte payload. One concern regarding the length of sequence number is that two bytes allow the storage of only 2^{16} different numbers. Longer sequence numbers, however, would result in increased overhead of ATM usage due to the decrease in payload size. We resolve this problem by recycling sequence numbers. That is, when sequence number reaches the

⁵ Actually, AAL1 and AAL3/4 cells contain a 4-bit sequence number field; and AAL2 cells, a 1-bit sequence number. However, these fields are not long enough to be used for cell retransmission.

upper bound, i.e. 2^{16} , it will start from 0 again. The mechanism for supporting recycling of sequence numbers is implemented in conjunction with the message stability detection scheme discussed in section 3.3.

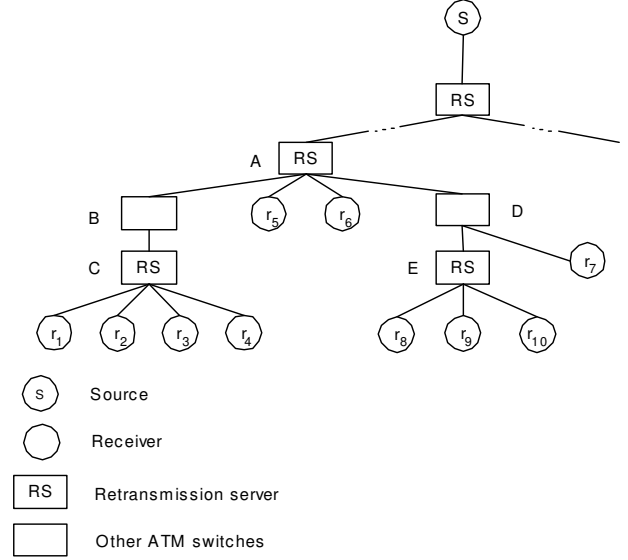


Figure 1: Part of the routing tree of an example multicast group with designated RSs

3.2 Request and Retransmission Algorithm

The request and retransmission algorithm uses local recovery: a receiver requests lost data from its retransmission server (RS), and the RS retransmits the requested data. RSs are ATM switches selected and organized hierarchically based on the existing ATM-established connection, i.e. the physical routing tree. A simple example is given in Figure 1, that shows part of the routing tree of a multicast connection with designated RSs. The terminal equipment connected to the sender is also a RS, and called the *root RS*. It buffers data from the sender for retransmission until that data has been received safely by all the receivers. For recovery, a receiver communicates only with its direct RS; a RS communicates only with its parent RS. For a specific multicast tree, the more RSs selected, the faster recovery and the less exposure. However, more RSs incur more overheads at these switches. Therefore, there is a trade-off between the number of RSs selected, and the recovery latency and exposure.

The basic request and retransmission algorithm is as follows. When a receiver detects a gap in the sequence of

data cells it receives, it sends a request⁶ upstream to its RS. Similarly, when a RS receives a data cell from its parent, it checks the sequence number to detect cell loss, and stores the cell into the buffer for retransmission. If there is a loss, the RS sends a request to its parent RS asking for the missing cell(s).

When a RS receives a request, if it has a copy of the requested cell in the buffer, it sends that cell along the link from which the request came, and discards the request. If the RS has also lost the requested cell, it simply discards the child's request because the RS must have sent a request of its own upstream, or will detect the loss soon and send a request.

When a RS receives a repair data cell, if the RS has been requesting and waiting for this cell, it stores the cell in the buffer, and multicasts the cell to its subtree. If the RS had received the cell safely before, it simply discards the repair cell to avoid the exposure problem. Thus exposure, if any, is limited to one level of descendants in the recovery tree. For example, in Figure 1, receiver r_7 detects a loss, and sends a request to its parent RS, node A . A sends the repair cell to switch D which then multicasts the cell to its subtree. Note that retransmission server E also receives the repair cell although E did not request it. E had received this cell safely before, and thus discards the repair cell now. The repair cell will not reach the children of E , namely receivers r_8 , r_9 and r_{10} .

It is recommended to select switches with high fan-out degrees to be RSs in order to minimize exposure. For instance, in Figure 1, either B or C could be the RS of r_1 , r_2 , r_3 and r_4 . If B were chosen, a request from a child receiver, say r_4 , would cause the repair cell to be sent by B to C , and then multicast by C to r_1 , r_2 , and r_3 as well. However, r_1 , r_2 , and r_3 did not request this cell. On the other hand, if C were the RS, C would send the repair cell along the link (C, r_4) for it to reach the receiver r_4 only. Therefore, C is a better choice for a RS.

3.3 Message Stability Detection

Message stability detection is very important to minimize storage overheads at RSs. Buffers for cells no longer

⁶ To simplify the discussion, we assume each request is for one lost cell. In practice, a receiver may ask for multiple data cells, or a block of cells in one request to conserve network bandwidth. Similar arguments are applicable to the case of requests for multiple cells.

needed by the children of a RS should be freed as soon as possible, since each RS is given a buffer of only limited size to serve an ATM multicast connection.

To implement message stability detection, we use a feedback control mechanism similar to ABR explicit rate control. Two new types of control cells are created: *forward stability detection* (FSD) and *backward stability detection* (BSD) cells. The source periodically multicasts FSD cells to the receivers. The frequency of sending out FSD cells depends on the source's transmission rate. Upon receiving a FSD cell, a receiver or a RS (except the root RS) returns a BSD cell recording the range of cells it has received safely so far. Every RS consolidates these values from its children in order to release buffers of cells no longer needed by the children.

We define the following variables, which every RS node (except the root RS) needs for message stability detection:

- *SafelyReceived*: This node has received *all* cells with sequence numbers less than or equal to *SafelyReceived*.
- *MinBuffered*: All cells with sequence numbers less than *MinBuffered* have been received safely by *all* the direct children of this node.
- *SubtreeSafelyReceived*: All cells with sequence numbers less than or equal to *SubtreeSafelyReceived* have been received safely by all the receivers in the *subtree* rooted at this RS node.
- *MaxReceived*: Data cell with the highest sequence number this node has received.
- *MaxSentByParent*: Data cell with the highest sequence number the parent of this node has multicast. The parent sends this information to the child using FSD cells. This value allows the child node to detect the loss of cells with sequence numbers in the range $(MaxReceived, MaxSentByParent]$.
- *MaxSentToChildren*: Data cell with the highest sequence number this node has multicast to its children. The node updates FSD cells with this value before forwarding the FSD cells to its children, who will use this value to update their *MaxSentByParent* variables.

The root RS requires only three variables: *MinBuffered*, *SubtreeSafelyReceived* and *MaxSentToChildren*. A receiver needs to maintain only three variables: *SafelyReceived*, *MaxReceived*, and *MaxSentByParent*. FSD cells contain a

field *MaxSentToChildren*; BSD cells, two fields: *SafelyReceived* and *SubtreeSafelyReceived*. The definitions of these fields are as given above. Their usage is discussed below.

When a receiver (or an RS) receives an FSD cell, it generates a BSD cell, records the value of *SafelyReceived* (and the value of *SubtreeSafelyReceived*) into the BSD cell, and sends the BSD to its parent. It also sets its variable *MaxSentByParent* to the value of field *MaxSentToChildren* of the FSD. An RS also records the value of its variable *MaxSentToChildren* into the FSD cell, and multicasts the FSD cell to its children. The pseudo-code is as follows:

Upon receiving an FSD cell:

```
Generate a BSD cell;
BSD.SafelyReceived = SafelyReceived;
IF this is a receiver
    BSD.SubtreeSafelyReceived = SafelyReceived;
ELSE IF this is a RS
    BSD.SubtreeSafelyReceived = SubtreeSafelyReceived;
Send the BSD upstream to the parent;
MaxSentByParent = FSD.MaxSentToChildren;
IF this is a RS {
    FSD.MaxSentToChildren = MaxSentToChildren;
    Multicast FSD cell to the children;
}
```

Every RS waits for BSD cells from its children. It consolidates values *SafelyReceived* from *all* of its children, and selects the minimum value. It then sets variable *MinBuffered* to this minimum value, and releases buffers of cells with sequence numbers up to *MinBuffered*. The RS also consolidates *SubtreeSafelyReceived* values from BSD cells in a similar way. To perform feedback consolidation, each RS *R* needs an array of booleans, one bit for each child of *R*. Bit *receivedBSD[i]* is set to true every time *R* receives a BSD cell from the child *i*. All elements of array *receivedBSD* are set to false every time *R* returns a BSD to its parent RS, i.e., at the end of every consolidation round. Let *MAX* denote the highest sequence number (e.g. $2^{16} - 1$), and *childMin* and *subtreeMin* be variables that are initialized to *MAX* at the beginning of every consolidation round. Following is the pseudo-code for processing BSD cells:

Upon receiving a BSD cell from child *i*:

```
receivedBSD[i] = true;
childMin = min(childMin, BSD.SafelyReceived);
subtreeMin = min(subtreeMin,
    BSD.SubtreeSafelyReceived);
```

Discard the BSD cell;

```
IF (for every child k, receivedBSD[k] == true) {
    /* End the current consolidation round */
    MinBuffered = childMin;
    Release buffers of cells with sequence numbers up to
    MinBuffered;
    SubtreeSafelyReceived = subtreeMin;
    /* Start a new consolidation round */
    for all k, receivedBSD[k] = false;
    childMin = MAX;
    subtreeMin = MAX;
}
```

The consolidation of *SafelyReceived* values from BSD cells informs a RS of the reception rate of its slowest direct child, and allows the RS to free buffer space as early as possible. The consolidation of *SubtreeSafelyReceived* values, on the other hand, lets the RS know about the reception rate of the slowest receiver in the subtree rooted at this RS. This information is propagated towards the root RS by means of BSD cells, which may then signal the sender to reduce or increase the transmission rate accordingly. Therefore, the message stability detection mechanism can be used as a type of *feedback rate control* for services other than ABR, adding extra help to network congestion control.

Note that during the consolidation process, RSs must employ a time-out mechanism to detect failed children so as to avoid indefinite waiting.

FSD and BSD cells are also used to support recycling of sequence numbers. To implement this, a one-bit field called *wrapAround* is added to FSD and BSD cells. When sequence numbers reaches the upper bound, i.e. 2^{16} , it will begin again from 0. In this case, the root RS sets *wrapAround* bit in FSD cells to true to let the other RSs and receivers know about a second data window. A *data window* is defined as a block of data with continuous sequence numbers to be buffered by a RS for retransmission. When there is no sequence number wrap-around, a RS maintains only one data window, which is that with sequence numbers from *MinBuffered* to *MaxSentToChildren* (the shaded area Figure 2a). When sequence number is wrapped around, the data window is split into two: one from *MinBuffered* to *MAX*, and the other from 0 to *MaxSentToChildren* (the shaded areas in Figure 2b). The second window [*MinBuffered*, *MAX*] disappears as soon as *MinBuffered* reaches *MAX*. That is, when all the children of the RS have received safely the cell with sequence number *MAX*, and all the data sent before this cell. In this case, the RS sets bit *wrapAround*

in FSD cells to false before multicasting the FSD cells to its children. The children will know that they will be working with only one data window, until the next wrap-around cycle.

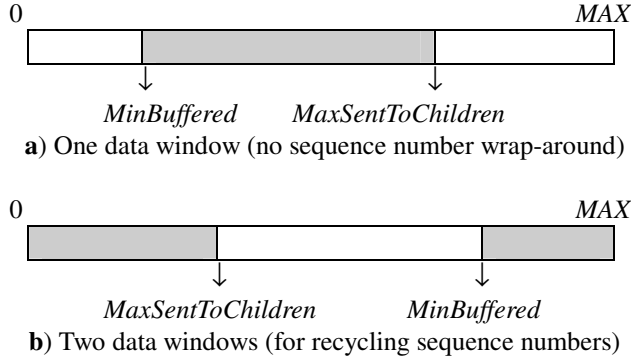


Figure 2: Data windows for recycling sequence numbers

The presence of two data windows affects the consolidation algorithm at RSs. Each RS now needs another variable called *numberOfWindows* to denote the number of data windows it is currently managing. The algorithm for processing BSD cells given above is extended as follows to account for the case of two data windows. As before, *childMin* and *subtreeMin* are initialized to *MAX* at the beginning of a consolidation round

Upon receiving a BSD cell from child *i*:

```
receivedBSD[i] = true;
IF ((numberOfWindows == 1) OR
    (numberOfWindows == 2 AND
     BSD.SafelyReceived > MinBuffered))
    childMin = min(childMin, BSD.SafelyReceived);
IF ((ssrWindows == 1) OR
    (ssrWindows == 2 AND
     BSD.SubtreeSafelyReceived > MaxSentToChildren))
    subtreeMin = min(subtreeMin,
                     BSD.SubtreeSafelyReceived);
Discard the BSD cell;
IF (for every child k, receivedBSD[k] == true) {
    /* End the current consolidation round */
    MinBuffered = childMin;
    Release buffers of cells with sequence numbers up to
    MinBuffered;
    SubtreeSafelyReceived = subtreeMin;
    IF (MinBuffered == MAX) numberOfWindows = 1;
    IF (SubtreeSafelyReceived == MAX) ssrWindows = 1;
    /* Start a new consolidation round */
    for all k, receivedBSD[k] = false;
    childMin = MAX;
```

```
subtreeMin = MAX;
}
```

In the above algorithm, if a RS currently has two data windows, it waits for feedback from each child indicating that the child has received all cells up to cell number *MAX* (i.e., *BSD.SafelyReceived* = *MAX*), or beyond cell number *MAX* (i.e., *BSD.SafelyReceived* < *MinBuffered*). After the RS has received this notification from *all* of its children, variable *MinBuffered* gets the value *MAX*, and the window [*MinBuffered*, *MAX*] disappears.

Variable *SubtreeSafelyReceived* is processed in a similar manner to *MinBuffered*, except that an additional variable *ssrWindows* is used to indicate the number of open windows in use by *SubtreeSafelyReceived*.

Note that the two data windows, when present, are not allowed to overlap at any instance. Overlapped windows at a RS would indicate that one or more receivers in the subtree rooted at that RS are not able to handle the current load. Therefore, to prevent window overlapping, the root RS should base on the value of its variable *SubtreeSafelyReceived* to request the sender to adjust the transmission rate accordingly. If window overlapping is about to happen, a RS may release buffers of cells that have not been acknowledged yet, preferably the least recent cells. It is likely that these cells have been received by the children of the RS, but not yet acknowledged.

3.4 Identifying New Data and Control Cells of Reliable Multicast Connections

Data cells of reliable multicast applications contain sequence numbers, and require special processing at RSs as described above. Therefore, reliable multicast data cells need to be tagged differently from other data cells such as those of unicast or unreliable multicast connections. The 3-bit payload type (PT) field in the ATM cell header is used to designate the type of information being carried in the 48-byte cell information field (Figure 3). We suggest the use of payload type (PT) value 111 to denote reliable multicast data cells since this value has not been used yet (see Table 1). Data cells with PT = 111 will bypass packet discard schemes, and be buffered at the RSs for retransmission.

GFC/VPI	VPI	
VPI	VCI	
VCI		
VCI	PT	CLP
HEC		

Note:

GFC	Generic Flow Control (4 bits)
VPI	Virtual Path Identifier (8 bits or 12 bits)
VCI	Virtual Channel Identifier (16 bits)
PT	Payload Type (3 bits)
CLP	Cell-Loss Priority (1 bit)
HEC	Header Error Control (8 bits)

Figure 3: Format of ATM cell header

PT Coding	Interpretation
000	User data cell, congestion not experienced, AAU = 0
001	User data cell, congestion not experienced, AAU = 1
010	User data cell, congestion experienced, AAU = 0
011	User data cell, congestion experienced, AAU = 1
100	Segment OAM F5 flow-related cell
101	End-to-end OAM F5 flow-related cell
110	Resource management cell
111	Unused; reserved for future functions

Table 1: Payload Type (PT) field coding

For request cells, FSD and BSD cells, there are two ways to implement them:

1. Use the same PT = 111 as data cells of reliable multicast connections. However, these control cells are denoted by the highest sequence number SN = 11...1, to be distinguished from data cells of reliable multicast. Within the payload of cells with PT = 111 and SN = 11...1, a coding field is used to denote request cells, FSD and BSD cells respectively.
2. Use PT = 110, as resource management (RM) cells of ABR flow control do (Table 1). Within the payload of (PT = 110) cells, a coding field is used to denote RM cells, request cells, FSD and BSD cells respectively.

4 Performance Evaluation

We assess the performance of REMAT protocol using simulation, a method commonly used for evaluating performance metrics related to ATM networks. We used the ATM network simulator version 4.1 distributed by the American National Institute for Standards and Technology (NIST) [17]. We added code to implement multicast routing, as well as data recovery at the transport layer and the ATM layer. For transport-layer reliable multicast, we implemented a tree-based protocol similar to LSM [12] since it is among high-performance tree-based protocols.

To make a fair comparison, we assume that a retransmission server in the ATM-layer protocol is also connected to a terminal equipment that can receive and process requests for lost packets, and retransmits those packets. The reason for the assumption is to show that whenever we have a transport-layer tree-based protocol, we can map the recovery tree at the transport layer to the ATM layer and obtain better performance.

4.1 Simulation Parameters

We have carried out experiments on two types of topology: binary trees, and WAN-like topologies. The binary trees have 16, 32, and 64 receivers, respectively. In each case, the sender is at the root of the binary tree. We ran experiments on several WAN-like configurations, and present the results from a typical configuration shown in Figure 4. In selecting the parameters for the experiments, we aimed at simulating the characteristics of WAN and IP over ATM. Following are the most important parameters that were used in the experiments:

- The senders are persistent sources, each transmitting a large data file of size 20 Mbytes.
- The network is overloaded by bursty sources to trigger cell loss in the multicast tree. The background traffic causes an approximate cell loss rate of 5%.
- All ATM links are STS-1 links with bandwidth of 51.84 Mbps.
- The distance between switches is 100 km. Terminal-switch distance is 1km.
- The packet size is 512 bytes. This is the maximum TCP segment size.

- For transport-layer reliable multicast, Early Packet Discard is used since it is one of the most commonly used packet discard algorithms and simple to implement.

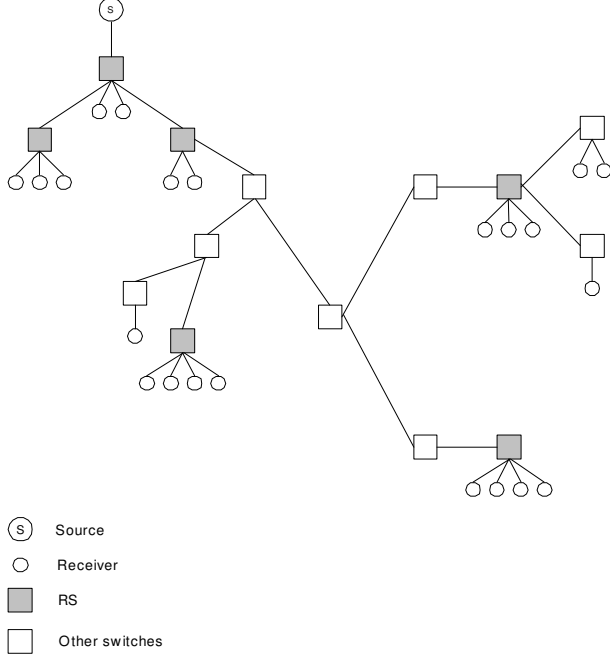


Figure 4: A WAN-like configuration

4.2 Simulation Results

In our experiments, we collect the following data:

1. *Average retransmission delay D* : For each packet lost, we measure the delay from the time the request is sent to the time the repair data is received. We then take the average of retransmission delays of all the retransmitted packets recorded by all the receivers. Redundant repair packets caused by exposure are not included in the computation.
2. *Connection throughput P* : Let S be the size of the data file to be transmitted by the source, and T be the time for the file to be received by *all* the receivers of the multicast group. T is basically the time for the whole multicast group to complete the transaction. Only then can the physical ATM connection be released to free resources. The connection throughput P is defined as $P = S/T$.

For every topology mentioned above, ATM-layer recovery and transport-layer recovery were simulated. For each topology, we measured the average retransmission delays D_{ATM} and $D_{transport}$, and the connection throughputs P_{ATM} and $P_{transport}$ from ATM-layer recovery and transport-layer recovery, respectively. We then calculated the average retransmission delay ratio $D_{ATM} / D_{transport}$ and the connection throughput ratio $P_{ATM} / P_{transport}$. The results are given in Table 2.

Topology	$D_{ATM} / D_{transport}$	$P_{ATM} / P_{transport}$
Binary tree 16	0.54	1.08
Binary tree 32	0.55	1.08
Binary tree 64	0.62	1.06
WAN in Figure 4	0.52	1.07

Table 2: Simulation results

The results listed on Table 2 show that the average retransmission delay ratios (ARDRs) given by ATM-layer recovery are almost half of those given by transport-layer recovery respectively.

In the experiments with binary trees of 16 and 32 receivers, we let the distance between a RS and its child receivers be two hops, and each RS serve four receivers, as shown in Figure 5. We observe that the ARDRs from the binary trees of 16 and 32 receivers are almost equal. The reason is that both trees have the same RS-receiver distance, and each RS serves the same number of children.

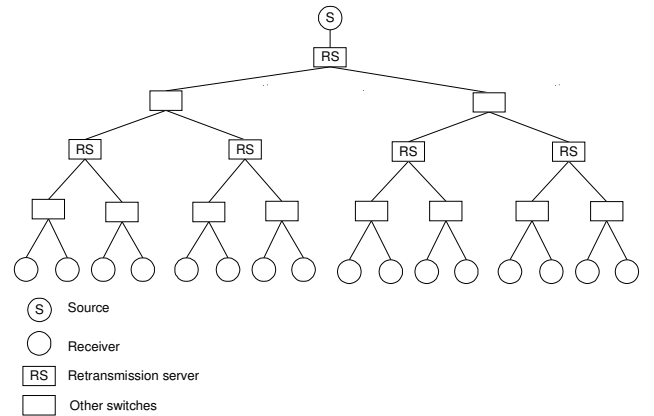


Figure 5: Binary tree with 16 receivers

In the case of binary tree with 64 receivers, we moved the RSs one hop up the binary tree, far away from the receivers. We see that the ARDR is higher than those of the binary trees with 16 and 32 receivers. The reason is that packet processing time is a major component in the retransmission delay of transport-layer recovery due to both request and repair packets. As the recovery distance increases, the latency increases but the packet processing time remains unchanged. Therefore, the overall delay of ATM-layer recovery increases at a higher rate than the overall delay of transport-layer recovery. As a result, the ARDR is higher as we move the RSs far away from the receivers.

For the WAN-like configuration in Figure 4, the ARDR is even lower than the ARDRs from the experiments with the binary trees, because we placed the RSs closer to the receivers.

Although ATM-layer recovery incurs a 4% overhead due to cell sequence numbers, it still offers better connection throughputs than transport-layer recovery under congestion causing a 5% cell loss rate in the experiments, as shown in Table 2. The reason is that only lost cells are retransmitted, instead of whole packets. Moreover, transport-layer recovery consumes time for segmenting, reassembling and transmitting request and repair packets, while ATM-layer recovery does not need those packets.

5 Conclusion and Future Work

Our new reliable multicast protocol proposes to move data loss recovery from the transport layer to the ATM layer. The purpose is to minimize recovery latency, which is essential to many important reliable multicast applications with strict time requirements. A second goal is to reduce unnecessary retransmission of repair packets at the transport-layer, which would worsen network condition under serious congestion. The new protocol uses local recovery based on the ATM routing tree. There is no repair implosion, and retransmission operates with minimal exposure, at the cell level. The message stability detection scheme can be used as a rate control mechanism to help alleviate congestion, and thus reduce recovery latency.

Experimental results show that the performance of ATM-layer recovery is better than that of transport-layer recovery in terms of average retransmission delays. ATM-layer recovery also offers better connection throughputs than transport-layer recovery under network congestion.

We have presented the basic features of the REMAT protocol. We are in the process of adding enhancements and fine-tuning the protocol. Future planned enhancements include handling lost request/repair cells efficiently, and developing the message stability detection scheme into a flow control mechanism. In addition, we will run more complex experiments in order to fine-tune the protocol parameters for optimal performance.

References

- [1] G.J. Armitage and K. M. Adams, "How Inefficient Is IP over ATM anyway?", *IEEE Network*, 1995.
- [2] B.J. Dempsey, J.C. Fenton, and A.C. Weaver, "The Multidriver: A Reliable Multicast Service Using the Xpress Transfer Protocol," *Proceedings of the 15th Conference on Local Computer Networks*, September 30 - October 3, 1990.
- [3] R. Talpade, and M.H. Ammar, "Single Connection Emulation (SCE): An Architecture for Providing a Reliable Multicast Transport Service," *Proceedings of International Conference on Distributed Computer System (ICDCS) '95*, October 1995.
- [4] B. Whetten, T. Montgomery, and S. Kaplan, "A High Performance Totally Ordered Multicast Protocol," *Theory and Practice in Distributed Systems*, Springer Verlag LNCS 938, 1995.
- [5] Jo-Mei Chang, and N.F. Maxemchuk, "Reliable Broadcast Protocols," *ACM Transactions on Computer Systems*, vol. 2, no. 3, August 1984.
- [6] S. Floyd, *et al.*, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing," *Proceedings of ACM SIGCOMM '95*, October 1995.
- [7] B. Sabata, M.J. Brown, and B.A. Denny, "Transport Protocol for Reliable Multicast: TRM," *Proceedings of International Conference on Networks*, January 1996.
- [8] S. Paul, K.K. Sabnani, J.C. Lin, and S. Bhattacharyya, "Reliable Multicast Transport Protocol (RMTP)," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 3, April 1997.

- [9] R. Yavatkar, J. Griffioen, and M. Sudan, "A Reliable Dissemination Protocol for Interactive Collaborative Applications," *Proceedings of ACM Multimedia '95*, 1995.
- [10] H.W. Holbrook, S.K. Singhal, and D.R. Cheriton, "Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation," *Proceedings of ACM SIGCOMM '95*, October 1995.
- [11] M. Hofmann, "Enabling Group Communication in Global Network," *Proceedings of Global Networking '97*, June 1997.
- [12] C. Papadopoulos G. Parulkar, and G. Varghese, "An Error Control Scheme for Large-Scale Multicast Applications," *Proceedings of IEEE INFOCOM '98*, 1998.
- [13] A.M. Costello and S. McCanne, "Search Party: Using Randomcast for Reliable Multicast with Local Recovery," *Proceedings of IEEE INFOCOM '99*, 1999.
- [14] B.N. Levine and J.J. Garcia-Luna-Aceves, "Improving Internet Multicast with Routing Labels," *Proceedings of International Conference on Network Protocols '97*, October 1997.
- [15] S.E. Deering, *et al.*, "The PIM Architecture for Wide-Area Multicast Routing," *IEEE/ACM Transactions on Networking*, vol. 4, no. 2, April 1996.
- [16] T. Ballardie, P. Francis, and J. Crowcroft, "Core Based Tree (CBT): An Architecture for Scalable Inter-Domain Multicast Routing," *Proceedings of ACM SIGCOMM '93*, September 1993.
- [17] The NIST ATM/HFC Network Simulator, Version 4.1, National Institute of Standards and Technology, http://w3.antd.nist.gov/Hsntg/prd_atm-sim.html