

Reliable Multicast Transport Protocol (RMTP)¹

Sanjoy Paul
Krishan K. Sabnani

Bell Laboratories
Holmdel, NJ 07733

John C. Lin

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

and

Supratik Bhattacharyya

Department of Computer Science
University of Massachusetts
Amherst, MA 01003

Abstract

This paper presents the design, implementation and performance of a reliable multicast transport protocol called RMTP. RMTP is based on a hierarchical structure in which receivers are grouped into local regions or domains and in each domain there is a special receiver called a Designated Receiver (DR) which is responsible for sending acknowledgments periodically to the sender, for processing acknowledgements from receivers in its domain and for retransmitting lost packets to the corresponding receivers. Since lost packets are recovered by local retransmissions as opposed to retransmissions from the original sender, end-to-end latency is significantly reduced, and the overall throughput is improved as well. Also, since only the DRs send their acknowledgments to the sender, instead of all receivers sending their acknowledgments to the sender, a single acknowledgement is generated per local region, and this prevents *acknowledgement implosion*. Receivers in RMTP send their acknowledgments to the DRs periodically, thereby simplifying error recovery. In addition, lost packets are recovered by selective repeat retransmissions, leading to improved throughput at the cost of minimal additional buffering at the receivers.

This paper also describes the implementation of RMTP and its performance on the Internet.

¹To appear in IEEE Journal on Selected Areas in Communications, special issue on Network Support for Multipoint Communication.

1 Introduction

Multicasting provides an efficient way of disseminating data from a sender to a group of receivers. Instead of sending a separate copy of the data to each individual receiver, the sender just sends a single copy to all the receivers. A multicast tree is set up in the network with the sender as the root node and the receivers as the leaf nodes. Data generated by the sender flows through the multicast tree, traversing each tree edge exactly once. However, distribution of data using the multicast tree in an unreliable network does not guarantee reliable delivery, which is the prime requirement for several important applications, such as, distribution of software, financial information, electronic newspapers, billing records, and medical images. Reliable multicast is also necessary in Distributed Interactive Simulation (DIS) environment, and in collaborative applications. Therefore, reliable multicasting is an important problem which needs to be addressed.

Several papers have addressed the issue of multicast routing [A84] [D88] [DC90] [BFC93] [KPP93] [Mo94] [DEF+94], but the design of a reliable multicast transport protocol in broadband packet-switched networks has only recently received attention [AFM92] [PSK94] [WMK95] [HSC95] [FJM+95] [PP95] [LP96].

Reliable multicast protocols are not new in the area of distributed and satellite broadcast systems [CM84] [GJ84] [SS85] [BJ87] [GS91] [APR93]. However, most of these protocols apply to local area networks and do not scale well in wide area networks, mainly because the entities involved in the protocol need to exchange several control messages for coordination purposes. In addition, they do not address fundamental issues of flow control, congestion avoidance, end-to-end latency, and propagation delays which play a critical role in wide area networks. Several new distributed systems have been built for group communication recently, namely, Totem [MMA+96] and Transis [DM96]. Totem [MMA+96] provides reliable totally ordered multicasting of messages based on which more complex distributed applications can be built. Transis [DM96] builds the framework for fault tolerant distributed systems by providing mechanisms for merging components of a partitioned network that operate autonomously and later become reconnected.

Both these systems assume the existence of multiple senders and try to impose a total ordering on delivery of packets. However, the reliable multicast transport protocol in this paper has been designed to operate at a more fundamental level where the objective is to deliver packets in ordered lossless manner from a single sender to all receivers. In other words, our protocol can potentially be used by Totem to provide reliable total ordering in a wide area packet-switched network. Other transaction-based group communication semantics like atomic multicast, permanence, and serializability can also be built using our reliable multicast transport protocol.

Multicasting is a very broad term and different multicasting applications have, in general, different requirements. For example, a real-time multipoint-to-multipoint multimedia multicasting application, such as, nationwide video conferencing, has very different requirements from a point-to-multipoint reliable data transfer multicasting application, such as, the distribution of software. Recently, researchers have demonstrated multicasting real-time data, such as real-time audion and video, over the Internet using the *MBone* [CD92][E94]. Since most real-time applications can tolerate some data loss but cannot tolerate the delay associated with retransmissions, they either accept some loss of data or use forward error correction for minimizing such loss. Multicasting of multimedia information has been recently receiving a great deal of attention [YM93][SM94][AS95]. However, the main objective of these multicast protocols is to guarantee quality of service by reducing end-to-end delay at the cost of reliability. In contrast, the objective of our protocol in this paper is to guarantee *reliability* achieving high throughput, maintaining low end-to-end delay. This is achieved by reducing unnecessary retransmissions by the sender. In addition, we adopt a novel technique of grouping receivers into local regions and generating a single acknowledgment per local region to avoid the acknowledgment implosion problem [R92] inherent in any reliable multicasting scheme. We also use the principle of periodic sending of state information from the receivers to the transmitter to avoid complex error-recovery procedures [NRS90]. Finally we use a selective repeat retransmission scheme to achieve high throughput.

In this paper, we describe our detailed experience

with the design and implementation of RMTP. The original work consisted of proposing three different multicast transport protocols, comparing them using simulation, and recommending one for reliable multicasting. In fact, the notion of local recovery using a designated receiver was proposed for the first time in the literature in [PSK94]. The details are reported in [PSK94], and a brief description is given in the Appendix. The recommended protocol was implemented and its performance, measured on the Internet, reported in [LP96]. In this paper, we have combined the ideas and results from [PSK94] and [LP96] to present a comprehensive picture of our efforts in designing RMTP.

RMTP is very general in the sense that it can be built on top of either virtual-circuit networks or datagram networks. The only service expected by the protocol from the underlying network is the establishment of a multicast tree from the sender to the receivers. For example, any multicast routing protocol, such as, DVMRP [DC90], PIM [DEF+94] or CBT [BFC93] can be used to set up this multicast tree. Further, ST-2 [PP92], RSVP [ZDE+93] or any other protocol can be used for reserving resources for the multicast tree. However, resource reservation is not really necessary for the proper functioning of RMTP. The function of RMTP is to deliver packets from the sender to the receivers in sequence along the multicast tree, independent of how the tree is created and resources are allocated. For example, RMTP can be implemented over Available Bit Rate (ABR) type service in ATM networks for reliable multicasting applications.

In this paper, we have addressed the design issues for RMTP in the internet environment. In particular, the notion of multi-level hierarchy using an internet-like advertisement mechanism is described, and issues related to flow control and late-joining receivers in an ongoing multicast session are dealt with extensively. In addition, a detailed description of the implementation using *MBone* [E94] technology in the Internet is also presented and performance measurements are included as well. Most of these ideas and results are taken from [LP96].

Rest of the paper is organized as follows. Section 2 discusses the network architecture and the assumptions made in the design of RMTP. RMTP is described in detail in Section 3. Implementation of RMTP is pre-

sented in section 4, and its performance measurements on the Internet are presented in Section 5. Comparison with related work is detailed in Section 6. Features and limitations of RMTP are summarized in section 7 followed by some conclusions.

2 Network Architecture and Assumptions

Let the senders and receivers be connected to the backbone network through local access switches² either directly or indirectly through access nodes³ (Figure 1).

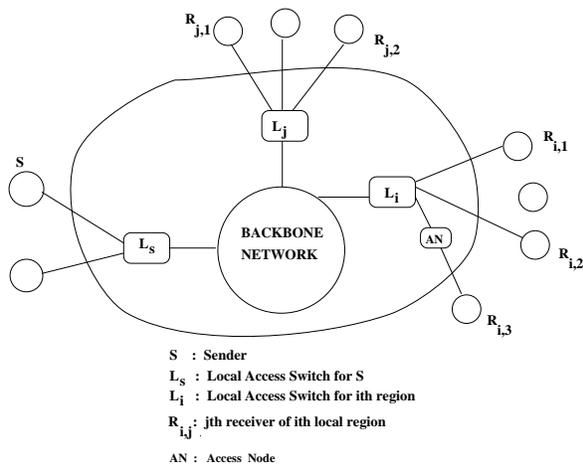


Figure 1: Model of the Network

The assumptions made in the protocol design are:

1. The receivers can be grouped into *local regions* based on their proximity in the network. For example, if a hierarchical addressing scheme like E.164 (which is very similar to the current telephone numbering system) is assumed, then receivers can be grouped into local regions based on area code. In an IP-network, receivers can be grouped into local regions by using the time-to-live (TTL) field of IP packets. More details on how the TTL field can be used are given in the next section.

²A local access switch can be thought of as a router in an IP-network.

³An access node is also a router in an IP-network.

2. A multicast tree, rooted at the sender S and spanning all the receivers, is set up at the network layer (ATM layer in the context of ATM networks). This is referred to as the *global multicast tree* in several parts of the paper to distinguish it from the *local multicast tree* which is a part of the global multicast tree⁴. The global multicast tree is shown by solid lines in Figure 2. Receivers in the local region served by L_i are denoted by $R_{i,j}$. Note that L_i denotes the local access switch for the i th. region and is *not* a receiver.

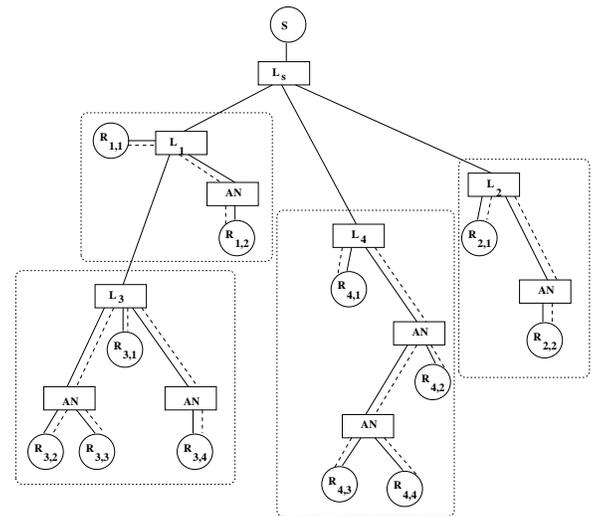


Figure 2: Global Multicast Tree rooted at S and Local Multicast Trees rooted at $R_{i,1}$'s

3. RMTP is described in this paper as a protocol for point-to-multipoint reliable multicast. Multipoint-to-multipoint reliable multicast is possible if multicast trees are set up for each sender.

3 Reliable Multicast Transport Protocol (RMTP)

RMTP provides sequenced, lossless delivery of bulk data from one sender to a group of receivers. The

⁴Note that the multicast tree is not assumed to be fixed. It may change dynamically as the network topology changes or as the membership of the multicast group changes. Although the multicast tree may change physically, there always exists a single logical multicast tree.

sender ensures reliable delivery by selectively retransmitting lost packets in response to the retransmission request of the receivers. If each receiver sends its status (ACK/NACK) all the way to the sender, it results in the throttling of the sender which is the well-known ack-implosion problem. In addition, if some receivers are located far away from the sender and the sender retransmits lost packets to these distant receivers, the end-to-end delay is significantly increased, and throughput is considerably reduced.

RMTP has been designed to alleviate the ack-implosion problem by using a tree-based hierarchical approach. The key idea in RMTP is to group receivers into local regions and to use a Designated Receiver (DR) as a representative of the local region. Although the sender multicasts every packet to all receivers using the global multicast tree, only the DRs send their *own* status to the sender indicating which packets they have received and which packets they have not received. The receivers in a local region send their status to the corresponding DR. Note that a DR *does not* consolidate status messages of the receivers in its local region., but uses these status messages to perform local retransmissions to the receivers, reducing end-to-end delay significantly. Thus the sender sees only the DRs and a DR sees only the receivers in its local region. Processing of status messages is distributed among the sender and the DRs, thereby avoiding the ack-implosion problem.

RMTP also supports multi-level hierarchy of local regions. In such a case, a DR sends its status to the DR least upstream from itself in the multicast tree and thus, the sender receives only as many status messages as there are DRs in the highest level of the multicast tree.

In Figure 2, receiver $R_{i,1}$ is chosen as the DR for the group of $R_{i,j}$'s, in the local region served by L_i . A local multicast tree, rooted at $R_{i,1}$, is defined as the portion of the global multicast tree spanning the $R_{i,j}$'s in the local region served by L_i . Local multicast trees are indicated by dashed lines in Figure 2.

3.1 Overview

This section presents the main ideas of RMTP assuming a two-level hierarchy as depicted in Figure 2. The extensions to multi-level hierarchy are straight-

forward. The protocol works as follows:

1. S multicasts a window of data packets to all receivers ($R_{i,j}$'s $\forall i, j$) using the global multicast tree. This multicast is termed a global multicast.
2. Each $R_{i,1}$ sends its own status to S in the form of status packets at periodic intervals. Each status packet contains information about which packets have been successfully received by $R_{i,1}$. Based on these status messages, S determines which packets are to be retransmitted. If the number of $R_{i,1}$'s requesting retransmission of a packet exceeds a certain threshold, the packet is multicast globally by S ; otherwise S unicasts the packet to the requesting $R_{i,1}$'s only.
3. Each $R_{i,j}$ ($j \neq 1$) sends its status to the corresponding $R_{i,1}$ at regular intervals. $R_{i,1}$ locally multicasts a packet if the number of $R_{i,j}$'s requesting its retransmission exceeds a threshold; otherwise the packet is unicast only to the $R_{i,j}$'s that requested its retransmission.
4. S multicasts new packets provided there is room in its send window.

3.2 RMTP Details

The sender in RMTP divides the data to be transmitted into fixed-size data packets, with the exception of the last one. A data packet is identified by packet type DATA, while type DATA_EOF identifies the last data packet. The sender assigns each data packet a sequence number, starting from 0. A receiver periodically sends ACK packets to the sender/DR. An ACK packet contains the lower end of receive window (L) and a fixed-length bit vector of receive window size indicating which packets are received and which packets are lost. Table 1 lists the packet types used in RMTP. Each of their functions will be described in the following subsections.

3.2.1 RMTP connection

An RMTP connection is identified by a pair of *end-points*: a source endpoint and a destination endpoint. The source endpoint consists of the sender's network address and a port number; the destination endpoint

Packet Types	
ACK	ACK packet
ACK_TXNOW	ACK - immediate transmission req.
DATA	Data packet
DATA_EOF	Last data packet
RESET	Packet to terminate a connection
RTT_MEASURE	Packet to measure round-trip time
RTT_ACK	ACK to RTT_MEASURE packet
SND_ACK_TOME	Packet for selecting an AP

Table 1: RMTP packet types

Connection Parameters	
W_r	receive window size in packets
W_s	send window size in packets
T_{daily}	delay after sending the last packet
T_{retr}	time interval to process retransmission requests
T_{rtt}	time interval to measure RTT
T_{sap}	time interval to send SND_ACK_TOME
T_{send}	time interval to send data packets
T_{ack}	time interval to send status packets
$Packet_Size$	data packet size in octets
$Cache_Size$	sender's in-memory data cache size
$CONG_{thresh}$	congestion avoidance threshold
$MCAST_{thresh}$	multicast retransmission threshold

Table 2: RMTP connection parameters

consists of the multicast group address and a port number. Each RMTP connection has a set of associated *connection parameters* (see Table 2). RMTP assumes that there is a *Session Manager*⁵ who is responsible for providing the sender and the receiver(s) with the associated connection parameters. RMTP uses default values for any connection parameter that is not explicitly given.

Once the Session Manager has provided the sender and receivers with the session information, receivers initialize the connection control block and remain in an unconnected state; the sender meanwhile starts transmitting data. On receiving a data packet from the sender, a receiver goes from the unconnected state to the connected state. In the connected state, receivers emit ACKs periodically, keeping the connection alive.

RMTP is designed based on the IP-Multicast philosophy in which the sender does not explicitly know who the receivers are. Receivers may join or leave a multicast session without informing the sender. Therefore the goal in RMTP is to provide reliable delivery to the *current* members of the multicast session. Since the sender does not keep an explicit list of receivers, termination of RMTP session is timer based. After the sender transmits the last data packet, it starts a timer that expires after T_{daily} seconds. (A DR also starts the timer when it has correctly received all the data packets.) When the timer expires, the sender deletes all state information associated with the connection (i.e., it deletes the connection's control block). Time interval T_{daily} is at least twice the lifetime of a packet in an internet. Any ACK from a receiver resets the timer to its initial value. A normal receiver deletes its connection control block and stops emitting ACKs when it has correctly received all data packets. A DR behaves like a normal receiver except that it deletes its connection control block only after the T_{daily} timer expires.

Since the time period between the transmission of consecutive ACKs from a receiver is much smaller than T_{daily} , the sender assumes that either all receivers have received every packet or something "exceptional" has happened. ACKs. Possible exceptional situations include: network partition and receivers voluntarily or involuntarily leaving the multi-

⁵Session Manager is not a part of RMTP transport protocol, but is used at the session layer to manage a given RMTP session.

cast group. RMTP assumes that the Session Manager is responsible for detecting such situations and taking necessary actions.

In addition to normal connection termination, RESET packets can be used to terminate connections. For example, when RMTP detects that the sending application has aborted before data transfer is complete, it uses RESET to inform all the receivers to close the connection.

3.2.2 RMTP Entities

RMTP has three main entities: (1) Sender, (2) Receiver and (3) Designated Receiver. A block diagram description of each of these entities is given in Figure 3. We describe the major components of these entities below.

The Sender entity has a controller component called T_CONTROLLER, which decides whether the sender should be transmitting new packets (using the Tx component), retransmitting lost packets (using the RTx component), or sending messages advertising itself as an ACK Processor⁶ (using the AP_A component and SEND_ACK_TOME message). There is another component called STATUS_PROCESSOR, which processes ACKs (status) from receivers and updates relevant data structures.

Also, note that there are several timer components: T_Send, T_Retx, and T_Sap in the Sender entity, to inform the controller about whether the Tx component, the RTx component or the AP_A component should be activated. Timer T_Dally is used for terminating a connection.

The Receiver entity also has a controller component called R_CONTROLLER which decides whether the receiver should be delivering data to the receiving application (using the R component), sending ACK messages (using the AS component), or sending RTT_measure packets (using the RTT component) to dynamically compute the round-trip time (RTT) between itself and its corresponding ACK Processor.

Note that there are two timer components: (1) T_Ack and (2) T_Rtt to inform the controller as to

⁶An ACK Processor (AP) for a receiver is the DR (or sender) to which the receiver sends its ACKs and on which it depends for retransmission of lost packets.

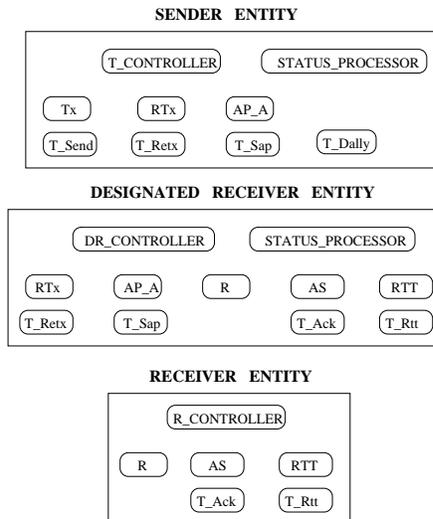


Figure 3: Block Diagram of RMTP

whether the AS or the RTT component should be activated. The component R is not timer driven. It is activated asynchronously whenever the receiving application asks for packets.

The DR entity is, in fact, a combination of the Sender entity and the Receiver entity.

Key functions performed by the components of each entity are described next.

3.2.3 Transmission

RMTP sender (in particular, the Tx component of sender entity in Figure 3) multicasts data packets at regular intervals defined by a configuration parameter T_{send} . The number of packets transmitted during each interval normally depends on the space available in send window⁷. The sender can at most transmit one full window of packets (W_s) during T_{send} , thereby limiting the sender's maximum transmission rate to $W_s * Packet_Size / T_{send}$. To set a multicast session's maximum data transmission rate, the Session Manager simply sets the parameters W_s , $Packet_Size$ and T_{send} accordingly. However, during network congestion, the sender is further limited by the congestion window⁸ during the same T_{send} interval.

⁷Note that space is created in send window when the lower end of the window slides after receiving acknowledgments from receivers.

⁸Refer to subsection 3.2.8.

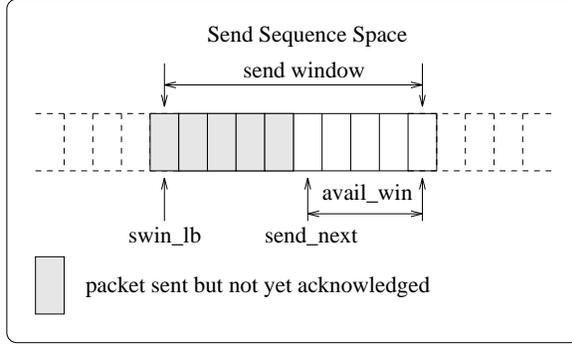


Figure 6: A sender’s send window and related variables

packets which are lost and hence need to be retransmitted. One or more receivers may miss the same packet. RMTP provides mechanisms for an AP to determine whether the lost packet should be retransmitted using unicast or multicast. Two parameters are used in the design for this purpose: T_{retx} , and $MCAST_{thresh}$, together with a *retransmission queue*. If an ACK contains retransmission requests, the sequence numbers of the requested packets are added to the retransmission queue. A retransmission queue element contains the sequence number of a packet to be retransmitted, a counter C that counts the number of receivers that have requested the packet, a table *AddrTab* that records the requesting receivers’ network addresses, and a pointer to the next queue element. At the end of interval T_{retx} , an AP (in particular, the RTx component of the sender/DR entity in Figure 3) processes each element in the retransmission queue. If C exceeds a threshold $MCAST_{thresh}$ ⁹, the AP delivers the packet using multicast; otherwise, the AP delivers the packet to each receiver in *AddrTab* using unicast.

The sender uses three variables, $swin_lb$, $send_next$, and $avail_win$ in the connection control block for managing the send window. As Figure 6 illustrates, variable $swin_lb$ records the lower bound of the send window, $send_next$ indicates the next sequence number to use when sending data packets, and $avail_win$ is the available window size for sending data. The sender increases $send_next$ and decreases $avail_win$ after sending data. When ACKs acknowledging the receipt of packets with sequence number

⁹The sender and DRs can have different $MCAST_{thresh}$ values.

$swin_lb$ are received, $swin_lb$ is increased and so is $avail_win$.

In order to determine how many new packets must be transmitted in the next send interval, the sender computes the smallest L (L_{min}) among those L values of ACKs received during T_{send} . If L_{min} is greater than $swin_lb$, it increases $avail_win$ by $(L_{min} - swin_lb)$ and sets $swin_lb$ to L_{min} . Value of $swin_lb$ is never decreased. If a receiver falls behind, and sends ACKs with values of L lower than $swin_lb$, those ACKs will be ignored. Eventually, however, the lagging receiver will send special ACKs called ACK_TXNOW (described in the next subsection) which will trigger retransmissions from a DR/sender.

3.2.6 Late Joining Receivers

Since RMTP allows receivers to join any time during an ongoing session, a receiver joining late will need to catch up with the rest. In addition, some receivers may temporarily fall behind because of various reasons such as network congestion or even network partition. There are two features in RMTP which together provide the functionality of allowing lagging receivers to catch up with the rest: (1) immediate transmission request and (2) data cache in the sender and the DRs.

Immediate Transmission Request

When a receiver joins late, it receives packets being multicast by the sender at that time, and by looking at the sequence number of those packets, it can immediately find out that it has missed earlier packets. At that instant, it uses an ACK_TXNOW packet to request its AP for immediate transmission of earlier packets. An ACK_TXNOW packet differs from an ACK packet only in the packet type field. When an AP receives an ACK_TXNOW packet from a receiver R , it checks bit vector V , and immediately transmits the missed packet(s) to R using unicast.

Data Cache

RMTP allows receivers to join an ongoing session at any time and still receive the entire data reliably. However, this flexibility does not come without a price. In order to provide this feature, the senders and the DRs in RMTP need to buffer the entire file during

the session. This allows receivers to request for the retransmission of any transmitted data from the corresponding AP. A two-level caching mechanism is used in RMTP. The most recent *Cache_Size* packets of data are cached in memory, and the rest are stored in disk.

3.2.7 Flow Control

A simple window-based flow control mechanism is not adequate in a reliable multicast transport protocol in the internet environment. The main reason is that in the internet multicast model, receivers can join or leave a multicast session without informing the sender. Thus a sender does not know who the receivers are at any instant during the lifetime of a multicast session.

Therefore if we want to design a transport-level protocol to ensure guaranteed delivery of data packets to all the *current* members of a multicast session, without explicitly knowing the members, a different technique for flow control is needed. Note that if RMTP used a simple window-based flow control mechanism, then the sender would have to know if all the DRs in level 1 have received the packets before the window is advanced. However, the sender may not know how many level 1 DRs are there, because the underlying multicast tree can change and either new DRs may be added to the multicast tree dynamically or old DRs may leave/fail.

In order to deal with this situation, the sender operates in a cycle. The sender transmits a window full of new packets in the first cycle and in the beginning of the next cycle, it updates the send window and transmits as many new packets as there is room for in its send window. The window update is done as follows. Instead of making sure that each level 1 DR has received the packets, the sender makes sure that all the DRs, that have sent status messages within a given interval of time, have successfully received the relevant packets before advancing the lower end of its send window. Note that the advancement of send window does not mean that the sender discards the packets outside the window. The packets are still kept in a cache to respond to retransmission requests. In addition, note that the sender never transmits more than a full window of packets during a fixed interval, thereby limiting the maximum transmission rate to

$W_s * Packet_Size / T_{send}$. This scheme of flow control can thus be referred to as rate-based windowed flow control. More details can be found in [PSLB96].

3.2.8 Congestion Avoidance

RMTP provides mechanisms to avoid flooding an already congested network with new packets, without making the situation even worse. The scheme used in RMTP for detecting congestion is described below.

RMTP uses retransmission requests from receivers as an indication of possible network congestion [J86, J88]. The sender uses a congestion window *cong_win* to reduce data transmission rate when experiencing congestion. During T_{send} , the sender computes the number of ACKs, N , with retransmission request. If N exceeds a threshold, $CONG_{thresh}$, it sets *cong_win* to 1. Since the sender always computes a *usable* send window as $Min(avail_win, cong_win)$, setting *cong_win* to 1 reduces data transmission rate to at most one data packet per T_{send} if *avail_win* is nonzero¹⁰. If N does not exceed $CONG_{thresh}$ during T_{send} , the sender increases *cong_win* by 1 until *cong_win* reaches W_s ¹¹. The procedure of setting *cong_win* to 1 and linearly increasing *cong_win* is referred to as *slow-start*, and is used in TCP implementation. The sender begins with a slow-start to wait for the ACKs from far away receivers to arrive.

3.2.9 Choice of Designated Receivers and formation of Local Regions

RMTP assumes that there is some information about the approximate location of receivers and based on that information, either some receivers or some servers are chosen as designated receivers (DRs). Although specific machines are chosen to act as DRs, the choice of an AP for a given local region is done dynamically. The basic idea is outlined below.

¹⁰Note that on detecting congestion in the network, it is possible to set the congestion window to one-half the size of current send window, instead of setting it to one as described here. We have not explored this possibility in details.

¹¹If the sender and all the receivers are located in an environment in which the sender's maximum data rate is unlikely to cause congestion, one can bypass RMTP's congestion avoidance scheme by setting $CONG_{thresh}$ to " ∞ ."

Each DR as well as the sender periodically sends a special packet, called the SEND_ACK_TOME packet, in which the time-to-live (TTL) field is set to a pre-determined value (say 64), using the multicast tree down to each receiver. Thus, if there are several DRs along a given path from the sender to a given receiver, the receiver will receive several SEND_ACK_TOME packets, one from each DR. However, since the TTL value of an IP datagram gets decremented by one at each hop of the network, the closer a DR is to a given receiver, the higher is the TTL value in the corresponding SEND_ACK_TOME packet. Therefore, if each receiver chooses the DR, whose SEND_ACK_TOME packet has the largest TTL value, it will have chosen the DR nearest to it in terms of number of hops. Effectively, a local region will be defined around each DR.

This approach gives us several benefits in terms of robustness and multiple levels of hierarchy. First of all, if the DR, selected by a set of receivers as their AP, fails, then the same set of receivers will choose the DR least upstream from the failed DR, as their new AP. This is because SEND_ACK_TOME packets from the failed DR will no longer arrive at the receivers and the SEND_ACK_TOME packet from the DR least upstream from the failed DR will have the largest TTL value. This leads to the dynamic selection of AP for a given set of receivers.

3.2.10 Multi-level Hierarchy in RMTP

RMTP has been described earlier as a two-tier system in which the sender multicasts to all receivers and DRs; and DRs retransmit lost packets to the receivers in their respective local regions. However, the limitations of a two-level hierarchy are obvious in terms of scalability and a multi-level hierarchy is desirable. The objective of this section is to describe how a multi-level hierarchy is obtained in RMTP with the help of the DRs sending SEND_ACK_TOME packets.

Recall that each DR periodically sends SEND_ACK_TOME packets along the multicast tree, and each receiver chooses the DR whose SEND_ACK_TOME packet has the largest TTL value. Moreover, note that each DR is also a receiver. Therefore, if each DR ignores its own SEND_ACK_TOME packets, it will choose the DR least upstream from it-

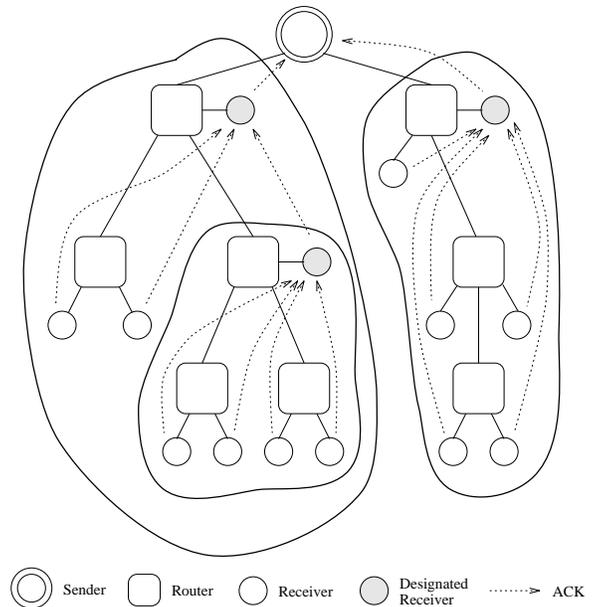


Figure 7: Multi-level Hierarchy of Designated Receivers

self as its DR and will send its status messages to that DR during the multicast session. Figure 7 illustrates the idea.

Effectively, if there are n DRs along a path from the sender to a group of receivers, and these DRs are different hop counts away from the receivers in question, there will be n local regions in an n -level hierarchy, such that the DR of the n th level¹² will send its status to the DR in level $n - 1$, a DR of level $n - 1$ will send its status to the DR in level $n - 2$, and so on, until the DR in level 1 sends its status to the sender (DR at level 0). That is, a DR at the i th level acts as a receiver for the $i - 1$ th level for all $i, i = n, \dots, 1$, where the 0th level refers to the global multicast tree rooted at the sender.

4 RMTP Implementation

RMTP uses Mbone technologies to deliver multicast packets. Mbone consists of a network of multicast capable routers and hosts. Mbone routers use IP tunnels to forward multicast packets to IP routers that cannot handle multicast packets. An Mbone router

¹²DR most upstream is level 1, and the level increases downstream along the multicast tree.

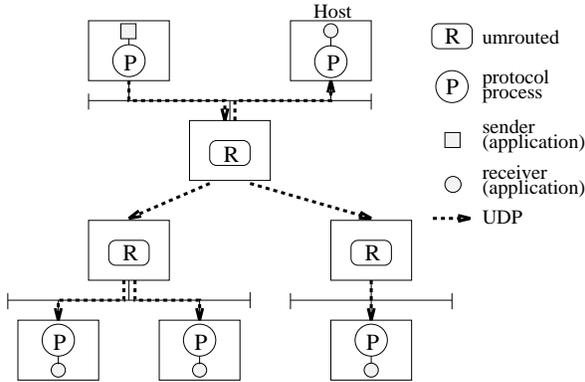


Figure 8: Multicast packet delivery from a sending application to a group of receiving applications using UDP

consists of two functional parts: a user-level process called `mrouterd` and a multicast kernel. An `mrouterd` exchanges routing information with neighboring `mrouterd`s to establish a routing data structure in the multicast kernel. The multicast kernel then uses the routing data structure to forward multicast packets. To deliver multicast packets to receivers on a local subnet, an MBone router uses data-link layer multicasting (e.g., Ethernet multicasting).

To make prototyping faster and debugging easier, we implemented multicast packet forwarding and RMTP protocol processing at user level. We modified `mrouterd` to incorporate the routing functions of a multicast kernel. (We refer to the modified `mrouterd` as `umrouterd`.) Communications among `umrouterd`s are via User Datagram Protocol (UDP) [P80]. Thus, multicast packets travel on UDP-tunnels among `umrouterd`s. By executing `umrouterd`, a host with unicast kernel becomes a user level multicast router.

A user-level *protocol process* implements the RMTP protocol. Application-level receivers and senders use UDP to communicate with the RMTP protocol process. To deliver multicast packets to protocol processes on a local subnet, a `umrouterd` uses UDP unicast instead of data-link multicast (see Figure 8).

A protocol process uses a configuration file to learn about the location of the `umrouterd` that handles its multicasting requests. When a protocol process wishes to join a multicast group, it sends an Internet Group Management Protocol (IGMP) [D89]

W_s	15	packets
W_r	32	packets
T_{send}	600	milli-second
T_{retx}	300	milli-second
T_{rtt}	1	second
T_{sap}	3	seconds
T_{daily}	250	seconds
$Packet_Size$	512	octets
$Cache_Size$	128	packets
$CONG_{thresh}$	0	
$MCAST_{thresh}$	3	

Table 3: Connection parameters used

Host Membership Report packet via UDP to its `umrouterd`. The Host Membership Report message requires an acknowledgment from the `umrouterd`. Thus, a `umrouterd` builds a list of protocol processes' host addresses that it handles. A `umrouterd` periodically sends an IGMP Host Membership Query message to each protocol process it handles using UDP unicast. Note that protocol processes and `umrouterd`s do not follow the IGMP protocol standards to obtain multicast group membership information because they encapsulate IGMP messages in UDP and do not use data-link multicast. In essence, we built a multicast delivery system at user level using MBone technologies.

5 Measurements on the Internet

We measured the prototype implementation's performance with 18 receivers located at 5 geographic areas. Figure 9 shows the network configuration used. We implemented a simplified version of the Session Manager (SM) and its clients. Each receiving host executes the client process and the protocol process in the background, and the SM uses TCP to transport session-related information (e.g., session ID, connection parameters) to each client. Upon receiving the information, the client process invokes an application-level receiver process and informs the protocol process about the session information. Each client reports back to the SM when the application-level receiver process is ready. SM starts the sender when all the application-level receiver processes are ready.

Table 3 shows the connection parameters assigned

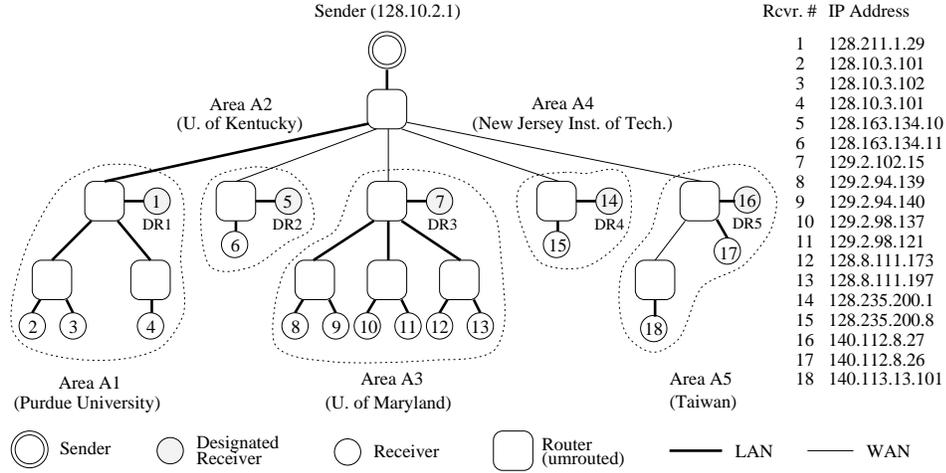


Figure 9: Network configuration used for measuring RMTP's performance

by the SM. A maximum data rate of 100 Kbits/second (Kbps) is chosen to avoid overloading the Internet links of the test sites. The $CONG_{thresh}$ is set to 0 so that the sender invokes slow-start whenever it receives a retransmission request from a receiver. DRs are chosen by using a configuration file. Note that the sender only processes the ACKs from the DRs.

We conducted 10 experiments. Each experiment consists of 3 measurements of multicast file transfer in different network environments — M1: the sender multicasts to area A1, a LAN environment; M2: the sender multicasts to areas A1 through A4, a WAN environment; M3: the sender multicasts to all areas, a WAN environment including an international link with 512 Kbps bandwidth. For each measurement, the sender reads a 1 megabyte file from file system and multicasts it to the receivers. Receivers store the received data in a file for integrity check. Each receiver computes throughput independently after successful reception of the file. We also measured, in each area, the total number of retransmitted packets and duplicate packets by examining the log files created by the sending or receiving protocol processes.

All the experiments were conducted between January 25 and January 28, 1995. The first 3 experiments were conducted between 9:00 and 12:00 EST; the second 3 experiments were conducted between 12:00 and 17:00 EST; and the rest were conducted between 21:00 and 24:00 EST. The hosts used in the experiments are all workstation-class computers (e.g. Sun IPC, Sun

No.	Throughput (Kb/sec)			Retransmissions (%)		# Slow Start
	min.	avg.	max.	sender	DR1	
1	90.33	90.35	90.38	0.00	0.00	1
2	91.33	91.37	91.38	0.00	0.05	1
3	90.62	90.64	90.65	0.00	0.00	1
4	81.38	81.40	81.41	0.00	0.00	1
5	73.05	73.08	73.10	0.00	1.37	1
6	86.92	86.93	86.95	0.00	0.00	1
7	91.78	91.78	91.79	0.00	0.00	1
8	92.26	92.29	92.31	0.00	0.00	1
9	91.62	91.65	91.68	0.00	0.00	1
10	91.07	91.22	91.31	0.00	0.00	1

Mean throughput: 88.07 Kb/sec.
* No duplicate reception observed.

Table 4: Results of multicasting to area A1

IPX, Sun Sparc10). The experiments were conducted with the normal user processes running on them. No special treatments were given to the hosts running RMTP.

The results of the experiments are categorized by their measurement types (i.e., M1, M2, or M3). Tables 4 to 6 show the results. The average throughput is plotted in Figure 10. Since each receiver computes its own throughput independently, the Tables show the minimum, average, and maximum throughput among the throughput numbers reported by receivers. Note that the Tables report the *total* number of retransmitted data packets observed by each AP, and the *total* number of duplicate data packets observed in each area. Thus, the numbers depend on the number of receivers in each area. As described in subsection ??, a DR receives duplicate packets from router when it uses

No.	Throughput (Kb/sec)			Total # of Retransmissions (%)					Total # of Duplicates (%)				# Slow Start
	min.	avg.	max.	sender	DR1	DR2	DR3	DR4	A1	A2	A3	A4	
1	36.43	36.46	36.48	4.93	0.10	0.29	8.30	0.83	0.10	0.00	3.56	0.00	32
2	24.50	24.53	24.56	14.11	0.00	2.44	28.12	2.20	0.00	0.00	5.32	0.05	113
3	11.46	11.48	11.49	54.88	0.44	19.48	83.35	2.00	0.00	0.00	12.35	0.00	599
4	21.40	21.42	21.44	10.64	0.05	0.34	14.31	3.71	0.10	0.00	3.22	0.15	136
5	28.14	28.15	28.15	6.49	0.59	0.59	9.23	2.49	0.00	0.00	1.76	0.00	73
6	28.57	28.57	28.58	6.59	14.11	0.44	10.55	2.69	0.00	0.00	2.83	0.00	65
7	38.79	38.82	38.91	2.93	0.00	0.54	8.94	0.54	0.00	0.00	0.20	0.10	29
8	41.09	41.10	41.11	3.81	0.00	0.24	10.06	1.07	0.00	0.00	2.34	0.00	23
9	39.72	39.73	39.74	3.61	0.49	0.24	8.54	0.10	0.00	0.00	2.39	0.00	26
10	41.91	41.93	41.93	3.71	0.00	0.05	11.87	0.78	0.00	0.00	0.63	0.05	22

Mean throughput: 31.22 Kb/sec.

Table 5: Results of multicasting to areas A1, A2, A3, and A4.

No.	Throughput (Kb/sec)			Total # of Retransmissions (%)						Total # of Duplicates (%)					# Slow Start
	min.	avg.	max.	sender	DR1	DR2	DR3	DR4	DR5	A1	A2	A3	A4	A5	
1	20.81	20.81	20.82	19.29	0.88	0.34	17.09	1.66	18.56	0.00	0.00	1.03	0.00	0.15	161
2	21.27	21.34	21.36	16.80	0.24	0.98	14.36	2.20	15.43	0.29	0.00	1.90	0.00	0.15	144
3	20.67	20.71	20.72	18.55	0.15	0.20	17.58	1.22	19.53	0.00	0.00	1.46	0.00	0.10	153
4	18.17	18.22	18.23	20.41	0.10	0.54	29.88	1.03	24.12	0.00	0.00	1.03	0.00	0.05	212
5	18.27	18.85	18.97	23.54	0.68	0.20	21.29	2.20	20.90	0.00	0.00	3.03	0.00	0.10	206
6	25.47	25.53	25.55	13.48	1.03	0.10	10.55	2.39	14.26	0.00	0.15	0.88	0.20	0.00	100
7	16.62	16.62	16.63	24.76	0.05	0.00	14.70	0.68	35.21	0.00	0.00	1.42	0.00	0.00	264
8	17.52	17.57	17.58	27.00	0.15	0.20	10.55	0.73	35.30	0.00	0.00	0.83	0.00	0.10	232
9	19.28	19.30	19.30	22.02	0.10	0.05	8.25	0.68	29.79	0.00	0.00	1.17	0.00	0.00	197
10	20.67	20.83	20.89	15.77	0.00	0.20	7.96	0.20	24.66	0.00	0.00	2.73	0.00	0.05	153

Mean throughput: 19.98 Kb/sec.

Table 6: Results of multicasting to all areas

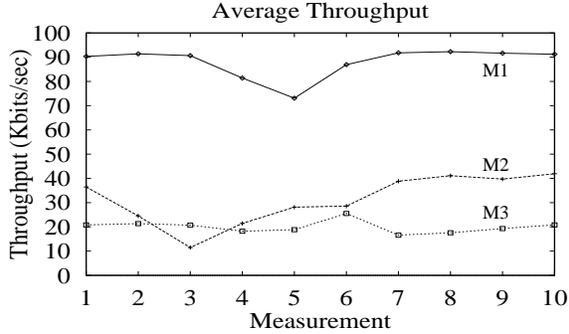


Figure 10: Average throughput among a group of receivers measured in various network environments

subtree multicasting to deliver retransmissions. The total number of duplicate data packets reported in the Tables does not include these duplicates. The numbers in percentage are calculated as the number of packets divided by 2048 (i.e., the size of the data file in number of packets).

From the results, we observe the following:

1. DRs play a major role, as expected, in caching received data, processing ACKs, and handling retransmissions. This is obvious from the “Total # of Retransmission” columns in Tables 4, 5 and 6. In particular, note that in Table 6, seven out of ten numbers in the column corresponding to DR5 are greater than those of the sender. This means that the DR in A5 (Taiwan) retransmitted more packets to its area than did the sender (in Purdue) to all areas. That means, if the DR were not there, all these retransmissions would have to be done by the sender. Effectively, the DRs shield the sender from handling local retransmission requests and provide faster response to the requests.
2. The small difference between the “max.” and the “min.” values of all the throughput measurements in Tables 4,5 and 6, indicates that receivers, regardless of their geographic location, take about the same time to correctly receive the file. This shows that RMTP is able to adapt to receivers in various network environments.
3. In a heterogeneous environment, slow receivers and links with low bandwidth limit RMTP’s performance. For example, with the same con-

nection parameters, RMTP achieved a mean throughput of 88.07 Kbps in M1 (a LAN environment), and a mean throughput of 19.98 Kbps in M3 (a WAN environment with a low bandwidth international link). On the one hand, it indicates RMTP has achieved its design decision of not over-running slow receivers and not wasting network bandwidth. On the other hand, it shows suboptimal throughput for fast receivers.

4. Low number of duplicate packets reported in areas A1, A2, A4 and A5 shows the effectiveness of RMTP’s T_{ack} calculation. The main cause for A3’s high number of duplicates is that DR3 uses multicast for delivering retransmitted packets. It can be explained by a simple example. Suppose that MC_{AST}_{thresh} is set to 3. Now if, 4 out of 6 receivers in A3 miss the same packet, DR3 will use subtree multicasting for retransmission of the missed packet. If all 6 receivers correctly receive the retransmission, two receivers will report duplicate reception.

6 Comparison with Existing Work

There is a wealth of literature on reliable multicasting, particularly in the context of distributed systems [CM84][KTHB89][BSS91]. Several new papers have also appeared in the recent literature [WMK95][PP95][HSC95][FJM+95], most of which focus on wide area networks.

[WMK95] describes the design of a reliable multicast protocol (RMP), which has significantly enhanced the work done by Chang and Maxemchuk in [CM84]. RMP provides different levels of QoS, namely unreliable, reliable, source ordered, totally ordered, K resilient, majority resilient and totally resilient. However, in order to provide these levels of QoS, the protocol requires exchange of several control messages among the members of a group. This is certainly possible in a local area network, but in a wide area network, exchanging these control messages will introduce high latency and the protocol design will not scale. In addition, RMP does not address several transport-level issues like flow control, congestion control, end-to-end latency, and redundant retransmission problems.

Our work is closely related to the Log-

Based Receiver-Reliable Multicast (LBRM) protocol [HSC95]. The distributed logging approach in LBRM is very similar to our hierarchical approach in RMTP which was first proposed in [PSK94]. However RMTP and LBRM differ significantly in details.

The Scalable Reliable Multicast (SRM) protocol by Floyd, Jacobson, MaCanne, Liu and Zhang takes a different approach from RMTP in recovering lost packets. In SRM, when a receiver detects missing data, it waits for a random time determined by its distance from the original source of the data, before it sends a repair request. Repair requests are multicast to the whole group just as regular data packets are. Thus, although a number of hosts may all miss the same packet, a host close to the point of failure is likely to timeout first and multicast the request. Other hosts that are also missing the same packet, hear that request and suppress their own request. This prevents a request implosion. Any host that has a copy of the requested data can answer a request. However, it will set a repair timer to a random value depending on its distance from the sender of the request message and multicast the repair when the timer goes off. Other hosts that had the packet and scheduled repairs will cancel their repair timers when they hear the multicast from the first host. This prevents a response implosion. This is different from the hierarchical approach in RMTP, in which a receiver requests retransmission of lost packets only from its DR and the DR retransmits the lost packets to the receiver. Thus the problems of request implosion or the repair implosion, that show up in SRM are eliminated in RMTP by design.

There is a problem with the recovery mechanism in SRM, normally referred to as the “crying baby” problem. If a single link to one member of the group has a high error rate, then all members of the multicast group will contend with a multicast request and one or more multicast responses. A member of the multicast group connected by a wireless link or a congested link will result in the “crying baby” problem. This situation is dealt with very efficiently in RMTP by using local recovery. More details on comparison can be found in [PSLB96].

7 Features and Limitations

7.1 Features

The main features of RMTP are summarized below:

1. Reliability: RMTP provides reliable delivery from a single sender to a group of receivers without knowing the exact identity of the receivers¹³. As described earlier in the paper, receivers send their status messages *periodically* to their Ack Processor (sender/DR) who retransmits any packets that are lost. Since ACKs are sent periodically by the receivers, even if an ACK gets lost, the sender/DR does not need to do anything special, because another ACK will be generated by the same receiver reflecting its updated status. Thus periodic sending of status messages provides an inherent fault tolerance to RMTP. If ACKs from the same receiver arrive out of order, the outdated ACK arriving later will be ignored by an Ack Processor (AP) if the value of L in the ACK is less than that of $swin_lb$ in the sender or the current L in the DR. Otherwise, the outdated ACK may lead to some redundant retransmissions. Since the value of $swin_lb$ at a sender or the value of L at a DR is monotonically increasing, correctness of the protocol is never compromised by out of order ACKs. If retransmitted packet itself is corrupted, it is detected by error checking codes just as in the case of UDP or TCP, and is treated like a lost packet. Thus the same packet will be requested for retransmission and will be eventually delivered. An RMTP receiver stops sending ACKs when it receives all packets successfully. Thus when the RMTP sender multicasts the last packet, and starts the T_{dally} timer, it expects to hear from a receiver/DR *only if* the receiver/DR does not receive every packet successfully. If an ACK gets lost, the receiver/DR will send a subsequent ACK when T_{ack} expires. As long as an ACK reaches the sender before T_{dally} expires, the sender will retransmit the lost packets, and restart T_{dally} . Since T_{dally} is a configurable parameter, its value can be chosen such

¹³Extensions to basic RMTP to provide guaranteed delivery to a known set of receivers are straightforward and are not included in this paper.

that the probability of a receiver not receiving the entire file correctly can be made arbitrarily small. Note that this problem will not exist if the sender exactly knows who the receivers are. RMTP has been extended to handle this case, but those extensions are beyond the scope of this paper.

2. Scalability: The hierarchical approach used in RMTP together with the design decision of not explicitly keeping track of receivers provide a high degree of scalability to RMTP. If some receivers in a multicast session are far from the original sender, the sender need not worry about them, because the corresponding DR will be responsible for both handling ACKs from and retransmitting lost packets to the faraway receivers. In addition, the state information kept at a sender is independent of the number of receivers, which is key to RMTP's scalability. The price RMTP pays for scalability is the additional cache at the sender and at each DR.
3. Heterogeneity: RMTP is able to handle receivers in heterogeneous network environments in an efficient manner. In particular, receivers in a relatively lossy network (say a wireless/congested network) can be made into a local region with a DR responsible for handling ACKs and retransmitting lost packets to the receivers in the region. Thus the effect of a lossy network can be confined to a small region without affecting other receivers of the same multicast session.

7.2 Limitations and Overheads

First of all, RMTP, as is designed today, requires the designated receivers to be chosen statically based on approximate location of the receivers. Ideally, the designated receivers must be determined dynamically as the receivers join and leave a multicast session. This is not really a limitation for applications where the set of receivers is known and specific receivers can be chosen as DRs. However, for applications with unknown set of receivers, RMTP would require some servers in the network to function as DRs in order to realize its full potential.

Secondly, as the receivers determine their DR based on the TTL value of the SEND_ACK_TOME packets,

it is possible for a large number of receivers to choose the same DR. This approach does not necessarily result in balancing of load among several DRs.

RMTP uses several periodic messages, such as, the status messages from the receivers and DRs; SEND_ACK_TOME packets from the sender and DRs; and the RTT_MEASURE packets from the receivers and DRs.

Typically, the status messages are sent by a receiver once per round-trip-delay between itself and the corresponding DR. This should not be considered an overhead, because each receiver has to send its status anyway and if a receiver takes an event-driven approach (as opposed to a periodic approach), in which it only sends NACKs when a loss is detected, the sender logic becomes more complex. For example, LBRM protocol [HSC95] takes this approach and hence the sender in LBRM needs to send periodic heartbeat messages to allow receivers to detect loss of packets quickly. Thus, sending of periodic status messages is not an overhead, rather it is a mechanism to simplify error recovery.

SEND_ACK_TOME packets are sent out periodically by the DRs and the sender in order to advertise that they can be used for error recovery by individual receivers. This is one of the two mechanisms that are currently being used for determining a local region. This technique is similar to router advertisements in [D91]. In the other approach, each receiver uses an expanding ring search to determine the nearest logging server [HSC95]. Thus the first technique puts the responsibility of defining a local region on the DRs, while the second relies on individual receivers to discover their corresponding logging servers. Sending an advertisement packet periodically is a standard mechanism used in the internet environment for router advertisements, foreign agent advertisements (mobile-IP), etc. Therefore, this is not an overhead one needs to be concerned about.

Finally, the receivers and the DRs send out RTT_MEASURE packets periodically. This is necessary for dynamically assessing the round-trip-delay to make the protocol operation efficient. The round-trip-time calculations are used by the receivers in determining how often they should be sending status messages. If this were not done dynamically, the protocol performance would be affected. There is a trade-off between

the performance of the protocol and the overhead in computing the round-trip-time dynamically. We have not investigated this trade-off yet.

8 Conclusion

This paper presented the complete design and implementation of a Reliable Multicast Transport Protocol called RMTP, and also provided performance measurements of the actual implementation on the Internet. The main contributions of the design include reducing the acknowledgment traffic by grouping receivers into local regions and generating a single acknowledgment per local region, and reducing end-to-end latency by performing local recovery. Contributions also include the extension of the two-level hierarchy to multi-level hierarchy of designated receivers in the internet environment. The idea of periodic advertisement by the designated receivers used in forming local regions is also new. Other contributions include the use of periodic status messages in the context of a reliable multicast transport protocol and the use of selective repeat retransmission mechanism to improve throughput. Although, the advantages of using periodic status messages are not explicitly discussed in this paper, it is known to reduce complex error handling mechanisms [NRS90]. In addition, this paper also presented experiences with a real protocol implementation on the Internet. In particular, the performance figures of RMTP implementation on the Internet justified the use of a hierarchical approach together with a DR in each local region as a smart mechanism for local recovery, and as a novel technique for achieving scalability in a heterogeneous network. The design decision of achieving reliability by building a hierarchy of local regions is supported by recent measurements done on Mbone by Yajnik et. al [YKT96] who show that most of the loss in Mbone happens at the local networks as opposed to in the backbone network. This suggests the use of a DR per local region at the points of departure from the backbone to deal with retransmissions of lost packets in an efficient manner. Finally, it has been conclusively shown by Levine et. al [LG96] that a tree-based reliable multicast transport protocol is the most scalable way of achieving reliability in a wide area packet-switched network. Based on these supporting arguments, RMTP is indeed a scal-

able, efficient, reliable multicast transport protocol.

Acknowledgment

Special thanks are due to Ramachandran Ramjee, Richard Buskens, Mischa Schwartz, Mark Karol, Bala Rajagopalan, and Thomas La Porta for their constructive comments and for their patience in going through this long paper. The authors would also like to thank Lixia Zhang, and the anonymous reviewers for their insightful comments.

References

- [A84] Lorenzo Aguilar, "Datagram Routing for Internet Multicasting", *ACM Computer Communications Review*, Vol. 14, No. 2, 1984, Pages 58-63.
- [AFM92] S. Armstrong, A. Freier and K. Marzullo, "RFC-1301: Multicast Transport Protocol," February 1992.
- [APR93] R. Aiello, E. Pagani and G. P. Rossi, "Design of a Reliable Multicast Protocol," *Proceedings of IEEE INFOCOM '93*, Pages 75-81, March 1992.
- [AS95] F. Adelstein, and M. Singhal, "Real-Time Causal Message Ordering in Multimedia Systems," *Proceedings of the 15th. ICDCS '95*, June 1995.
- [B89] R. Braden, "RFC-1122: Requirements for Internet Hosts - Communication Layers," October 1989.
- [BFC93] T. Ballardie, P. Francis and J. Crowcroft, "Core Based Trees (CBT): An Architecture for Scalable Inter-Domain Multicast Routing," *Proceedings of ACM SIGCOMM '93*, September 1993.
- [CD92] S. Casner and S. Deering, "First IETF Internet Audiocast," *ACM Computer Communications Review*, Vol.22, No. 3, July 1992.
- [CM84] Jo-Mei Chang and N.F. Maxemchuk, "Reliable Broadcast protocols," *ACM Transactions on Computer Systems*, Vol. 2, No. 3, August 1984, Pages 251-173.
- [BJ87] K. P. Birman and T. A. Joseph, "Reliable Communication in the Presence of Failures," *ACM Transactions on Computer Systems*, Vol. 5, No. 1, February 1987.

- [BSS91] K. P. Birman, A. Schiper and P. Stephenson, "Lightweight Causal and Atomic Group Multicast," *ACM Transactions on Computer Systems*, Vol.9, No.3, Pages 272-314, August 1991.
- [GS91] H. Garcia-Molina, and A. Spauster, "Ordered and Reliable Multicast Communication," *ACM Transactions on Computer Systems*, 9(3), Aug. 1991.
- [D88] Stephen E. Deering, "Multicast Routing in Inter Networks and Extended LANs," *ACM Computer Communications Review*, Vol. 19, No. 4, 1988, Pages 55-64.
- [D89] S. E. Deering, "RFC-1112: Host Extension for IP Multicasting," August, 1989.
- [D91] S. E. Deering, "RFC-1256: ICMP Router Discovery Messages," September, 1991.
- [DC90] S. E. Deering and D. R. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs," *ACM Transactions on Computer Systems*, Vol.8, No.2, Pages 85-110, May 1990.
- [DEF+94] S. E. Deering, D. Estrin, D. Farinacci, V. Jacobson, C-G Liu and L. Wei, "An Architecture for Wide-Area Multicast Routing," *Proceedings of ACM SIGCOMM '94*, Pages 126-135, October 1994.
- [DJNS93] B. T. Doshi, P. K. Johri, A. N. Netravali and K. K. Sabnani, "Error and Flow Control Performance of a High-Speed Protocol," *IEEE Transactions on Communications*, Pages 707-720, May 1993.
- [DM96] D. Dolev, and D. Malki, "The Transis Approach to High Availability Cluster Communication" *Communications of the ACM*, Vol.39, No.4, Pages 64-70, April 1996.
- [E94] H. Eriksson, "MBONE: The Multicast Backbone," *Communications of the ACM*, Vol. 37, No. 8, Pages 54-60, August 1994.
- [FJM+95] S. Floyd, V. Jacobson, S. McCanne, C-G. Liu and L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing," *Proceedings of ACM SIGCOMM '95*, Pages 342-356, October 1995.
- [GJ84] I. Gopal and J. Jaffe, "Point to Multipoint Communication over Broadcast Links," *IEEE Transactions on Communications*, Vol. COM-32, No. 9, September 1984.
- [HSC95] H.W. Holbrook, S.K. Singhal and D.R. Cheriton, "Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation," *Proceedings of ACM SIGCOMM '95*, Pages 328-341, October 1995.
- [J86] R. Jain, "A Timeout-Based Congestion Control Scheme for Window Flow-Controlled Networks," *IEEE Journal on Selected Areas in Communications*, Vol. SAC-4, No. 7, Pages 1162-1167, October 1986.
- [J88] V. Jacobson, "Congestion Avoidance and Control," *Proceedings of ACM SIGCOMM '88*, Pages 314-328, August 1988.
- [KPP93] V.P. Kompella, J.C. Pasquale, and G.C. Polyzos. Multicast routing for multimedia communication, *IEEE/ACM Trans. on Networking*, Vol. 1, No. 3, Pages 286-292, June 1993.
- [LG96] B.N. Levine, and J.J. Garcia-Luna-Aceves, "A Comparison of Known Classes of Reliable Multicast Transport Protocols," to appear in *Proceedings of International Conference on Network Protocols*, October 1996.
- [LP96] J.C. Lin and S. Paul, "RMTP: A Reliable Multicast Transport Protocol," *Proceedings of IEEE INFOCOM '96*, Pages 1414-1424, March 1996.
- [MMA+96] L.E. Moser, P.M. Melliar-Smith, D.A. Agarwal, R.K. Budhia, and C.C. Lingley-Papadopoulos, "Totem: A Fault-Tolerant Multicast Group Communication System," *Communications of the ACM*, Vol.39, No.4, Pages 54-63, April 1996.
- [Mo94] J. Moy. Multicast routing extensions for OSPF, *Comm. of the ACM* Vol. 37, No. 8, Pages 61-66, August 1994.
- [NRS90] A. N. Netravali, W. D. Roome and K. Sabnani, "Design and Implementation of a High-Speed Transport Protocol," *IEEE Transactions on Communications*, Vol.38. No.11, Pages 2010-2024, November 1990.
- [P80] J. Postel, "RFC-768: User Datagram Protocol," August 1980.
- [P81a] J. Postel, "RFC-791: Internet Protocol," September 1981.
- [P81b] J. Postel, "RFC-793: Transmission Control Protocol," September 1981.
- [PP92] C. Patridge and S. Pink, "An Implementation of the Revised Internet Stream Protocol (ST-2)," *Journal of Internetworking: Research and Experience*, Vol.4, No.1, March 1992.
- [PP95] C. Papadopoulos and G. Parulkar, "Implosion Control in Multipoint Transport Protocols,"

- Proceedings of IEEE Computer Communications Workshop*, September, 1995.
- [PSK94] S. Paul, K. K. Sabnani, and D. M. Kristol, "Multicast Transport Protocols for High Speed Networks," *Proceedings of International Conference on Network Protocols*, Pages 4-14, 1994.
- [PSLB96] S. Paul, K. K. Sabnani, J.C. Lin, and S. Bhattacharyya, "Reliable Multicast Transport Protocol (RMTP): A Detailed Report," Available from sanjoy@bell-labs.com.
- [R92] B. Rajagopalan, "Reliability and Scaling Issues in Multicast Communication," *Proceedings of ACM SIGCOMM '92*, Pages 188-198, September 1992.
- [SM94] N. Shacham, and J. S. Meditch, "An Algorithm for Optimal Multicast of Multimedia Streams," *Proceedings of IEEE INFOCOM '94*, Pages 856-864, June 1994.
- [SS85] K. Sabnani and M. Schwartz, "Multidestination Protocols for Satellite Broadcast Channels," *IEEE Transactions on Communications*, Vol. COM-33, No. 3, Pages 232-240, March 1985.
- [WMK95] B. Whetten, T. Montgomery, and S. Kaplan, "A High Performance Totally Ordered Multicast Protocol," *Theory and Practice in Distributed Systems K.P. Birman, F. Mattern, A. Schiper (Eds) Springer Verlag LNCS 938*.
- [YKT96] M. Yajnik, J. Kurose, and D. Towsley, "Packet Loss Correlation in the Mbone Multicast Network: Experimental Measurements and Markov Chain Models," submitted for publication.
- [YM93] R. Yavatkar, and L. Manoj, "Optimistic Approaches to Large-Scale Dissemination of Multimedia Information," *Proceedings of ACM MULTIMEDIA '93*, August 1993.
- [ZDE+93] L. Zhang, S. Deering, D. Estrin, S. Shenker and D. Zappala, "RSVP: A New Resource ReSerVation Protocol," *IEEE Network Magazine*, September 1993.

APPENDIX

In this section, we briefly mention the three protocols from which RMTP was chosen. The three protocols are: (1) Designated Status Protocol (DSP), (2) Consolidated Status Protocol (CSP), and (3) Combined Protocol (CP). Novelty of each of these protocols is in generating a single ACK from each local

Parameters	
Throughput	$CSP \leq DSP \leq CP$
End-to-end Delay	$DSP = CP \leq CSP$
Buffer Requirement	$CSP \leq DSP \leq CP$
Global Re-multicast Traffic	$CP \leq DSP \leq CSP$
Local Re-multicast Traffic	$DSP \leq CP$ Does not apply to CSP
Acknowledgment Traffic	$DSP = CSP = CP$
Processing Complexity	$CSP \leq DSP \leq CP$

region, thereby avoiding the ack-implosion problem. RMTP is derived from DSP and hence we skip its description here.

In CSP, each receiver sends its status to the corresponding local access switch L_i and the L_i combines status messages from all the receivers in its domain, and reports a packet loss to the sender if at least one of the receivers in its local region loses the packet. Eventually the sender retransmits the lost packets.

In CP, each receiver sends its status to the corresponding local access switch L_i and the L_i combines status messages from all the receivers in its domain, and reports a packet loss to the sender only if all the receivers in its local region lose the packet. A lost packet in a local region is retransmitted by any receiver which has received the packet.

The above table compares the performance of DSP, CSP, and CP.

Based on the performance of the protocols, we observe that CP has the best performance, followed by DSP and CSP. However, the improvement in performance is not without price. CP has substantially more overhead than the other protocols in terms of computing the DR dynamically, and communicating it to the members of a local region. DSP has the inherent simplicity of choosing the DRs right in the beginning of a session. In addition, performance of DSP is comparable to that of CP. Considering these factors, DSP becomes the protocol of choice. RMTP inherits the main ideas from DSP.