NASA-IVV-97-009

NASA IV&V Facility, Fairmont, West Virginia

High-Performance, Reliable Multicasting: Foundations for Future Internet Groupware Applications

John Callahan, Todd Montgomery, and Brian Whetten

July 14, 1997

This research was funded under cooperative Agreement #NCC 2-979 at the NASA/WVU Software Research Laboratory.

(LONG MANUSCRIPT)

High-Performance, Reliable Multicasting: Foundations for Future Internet Groupware Applications*

John Callahan, Todd Montgomery, Brian Whetten
{callahan,tmont}@cerc.wvu.edu, whetten@cs.berkeley.edu
NASA/West Virginia University Software IV&V Facility

January 9, 1996

Abstract

Network protocols that provide efficient, reliable, and totally-ordered message delivery to large numbers of users will be needed to support many future Internet applications. The Reliable Multicast Protocol (RMP) is implemented on top of IP multicast [14] to facilitate reliable transfer of data for replicated databases and groupware applications that will emerge on the Internet over the next decade. This paper explores some of the basic questions and applications of reliable multicasting in the context of the development and analysis of RMP.

1 Introduction

As the Internet continues to grow, new approaches are needed to provide large numbers of users with efficient, concurrent, on-demand access to information. In current approaches, like the client-server model used in the World-Wide-Web (WWW), information services are centralized and become bottlenecks under high demand. New protocols are needed to replicate data to multiple sites in order to distribute such demand. Replication enhances data availability and fault tolerance by providing alternative data sources under congestion and failure conditions. In addition to replicating databases and files, client applications will be required to "cache" more information locally and rely on notification from information providers of subsequent changes. New protocols for group applications can control the semantics, frequency, and granularity of notifications and updates at the application level [2].

Reliable broadcast and multicast protocols will play major roles in the development of large-scale replicated data systems [13]. Such protocols will be needed to maintain coherent copies of data at multiple sites in an efficient manner [10, 9]. In the past, however, reliable multicast protocols have had problems with performance, efficiency, and/or scalability. It has become a widespread belief that these are inherent problems with totally ordered reliable multicast protocols in general [27]. In reality, this misconception resulted from the fact that multicast had to be implemented as a series of unicasts to each destination. Recent developments, however, such as the IP Multicast [14] allow multicast datagrams to be routed efficiently to multiple destinations over an internetwork. In the case where all destinations are on the same LAN, one multicast packet to all destinations costs the same as a unicast packet to a single destination.

*This work is supported by NASA Grant NAG 5-2129 and NASA Cooperative Research Agreement NCCW-0040. More information pertaining to RMP can be found at `http://research.ivv.nasa.gov/projects/RMP/RMP.html`

This paper presents a brief overview of the details and potential applications of a new approach, called the the Reliable Multicast Protocol (RMP), that provides a scalable, totally ordered, reliable, atomic multicast service on top of an unreliable multicast datagram service such as IP Multicast. RMP is fully and symmetrically distributed so that no site bears an undue portion of the communication load. RMP provides a wide range of guarantees, from unreliable delivery to totally ordered delivery, to K-resilient, majority resilient, and totally resilient atomic delivery. These semantics are selectable on a per packet basis. RMP provides many communication options, including virtual synchrony, a publisher/subscriber model of message delivery, a client/server model of delivery, an implicit naming service, mutually exclusive handlers for messages, and mutually exclusive locks. It has commonly been held that a large performance penalty must be paid in order to implement total ordering. On SparcStation5's on a 1250 KB/sec Ethernet, RMP provides totally ordered packet delivery to one destination at 1070 KB/sec throughput and with 4.0 ms packet latency. The performance stays roughly constant independent of the number of destinations. For two or more destinations on a LAN, RMP provides higher throughput than any protocol that does not use multicast or broadcast [23, 24].

RMP provides a robust foundation for many data replication and groupware applications. In addition to replicated databases and file systems, current and potential applications of RMP include:

Distributed interaction simulations (DIS) . Distributed interactive simulation (DIS) environments consist of large numbers of autonomous and semi-autonomous agents (i.e., computer programs) that interact with one another to provide realistic scenarios for military training, planning, and strategic or tactical evaluation. DIS environments rely on local and wide-area networks as message handling systems to distribute information between these agents in a timely fashion. The overhead of message handling is substantial since the information exchanged between agents consists of large volumes of physical, geographical, and logistical data. Using RMP, a research group at the U.S. Army's Construction Engineering Research Laboratories has built a new DIS architecture called HPCADIS, that provides much higher performance than other DIS architectures.

Multicast URLs . We are designing enhanced WWW servers that can respond to *multicast URL* requests from WWW browsers. A multicast URL (i.e., `mhttp://...`) is a WWW resource that can be answered by any WWW server that provides the requested resource. Collisions between providers are prevented because multicast URLs can be implemented using RMP's multi-RPC feature. This feature provides remote procedure call access to groups of servers from non-member clients. A related system, called WEBCAST, has been developed using RMP at the University of Illinois to synchronize large numbers of WWW browsers in distributed learning applications.

Replicated objects . Distributed object models such as the Common Object Request Broker Architecture (CORBA) [16] will need to provide replication services between groups of objects in order to reduce contention. RMP does not provide a general enough framework to provide generic object replication services. To solve this problem, we are developing a *replicated objects layer* and *transaction model* on top of RMP to provide a standard framework that allows for customization of replications and object interoperability.

Wireless and Satellite Communications . Wireless and satellite media provide inherent support for broadcast and multicast transmission. Experiments with RMP are underway to analyze flow control, fault tolerance, and other customizations of RMP in such environments. Like multicasting in wired networks, wireless media provides an effective means for "pushing" data from servers to users in high demand applications like information delivery services [1].

RMP was developed at the NASA Independent Verification and Validation (IV&V) Facility in Fairmont, West Virginia as testbed project to explore new test and analysis techniques on complex, distributed software programs [3]. RMP has been extensively tested and analyzed to increase confidence in the correctness of the protocol specification and its implementation [7, 30]. The remainder of this paper describes some details of the protocol model, its limitations, features, and future research directions, but proofs of protocol correctness are beyond the scope of this paper.

2 Background

The basic RMP protocol provides what can be thought of as N-way virtual circuits, called groups, between sets of processes connected by a multicast medium. It is fully distributed, so that all processes play the same role in communication. While primarily using NACKs for error detection and retransmission, RMP provides true reliability and limits the necessary buffer space by passing a token around the members of a group.

RMP provides a wide range of reliability and ordering guarantees on packet delivery, selectable on a per packet basis. In addition to unreliable and reliable but unordered quality of service (QoS) levels, RMP can provide atomic, reliably delivery of packets ordered with respect to each source. It can also efficiently provide delivery of packets in both total and causal order, using causal ordering as defined in [21]. Totally ordered delivery also provides virtual synchrony, as first defined by the ISIS project [28]. Virtual synchrony guarantees that when new members join or leave a group these operations appear to be atomic, so that the sets of messages delivered before and after each membership change are consistent across all sites. Using K-resilient fault tolerance, RMP can provide total ordering and atomicity guarantees even in the face of site failures and partitions. For a set of packets with a resilience level of K, more than K members of a group have to simultaneously partition away or fail in order to have the possibility of violating the total ordering and atomicity guarantees. By setting K to a number larger than half the members of a ring and not allowing minority partitions to continue, total ordering, atomicity, and virtual synchrony can be guaranteed in the face of any set of arbitrary partitions and failures.

The basic RMP model of communication is a publisher/subscriber model based on textual group names. In the absence of network partitions, any member of a group (a subscriber) will receive all packets sent (published) to the group associated with that group name. RMP also provides a client/server model of communication, where the servers are members of a group and the clients are not members, but can communicate with the servers by sending packets to the group. These packets may be simply acknowledged after being delivered to the group with the requested QoS, or they may be responded to by a single member of the group. RMP uses handlers to guarantee that at most one member will respond to a data packet. Each data packet in RMP has an optional handler number associated with it. These correspond to a set of mutually exclusive handler locks which group members may hold. The group member who holds a given handler lock will be notified upon delivery of a data packet with this handler number that it is supposed to respond to the request. Handler locks are provided in a very efficient way, and can be used for any type of application that requires mutually exclusive locks shared among a group of communicating processes.

A common belief in the research community is that totally ordered reliable multicast protocols are inherently slow. This belief has come about in large part due to the experiences researchers have had with the early versions of ISIS, which for a long time was the only system of this type available. ISIS has since become much faster [5], but the misconception remains. Experience with RMP belies this concept. RMP was tested on 8 SparcStation5's on a 10 Mb/sec (1250 KB/sec) Ethernet. In this environment, the throughput to a single destination is 1070

KB/sec, or 86% of the network capacity. For group communication to any group of two or more destinations on a LAN, RMP exceeds not only the maximum throughput of TCP/IP, but any other possible non-multicast and non-broadcast algorithm. This is because both the packet latency and throughput of RMP stay roughly constant as the number of destinations increase, whereas the performance of other algorithms decreases linearly. For a group with 8 destinations, RMP has a 7.4 MB/sec aggregate throughput, which is 5.9 times the bandwidth of the supporting Ethernet. The throughput for RMP does not significantly change as a factor of the ordering guarantees, but the per packet latency does. A totally ordered packet will on average have a latency approximately twice that of an unordered or source ordered packet, and this increases for K-resilient packets. This QoS for latency tradeoff is fundamental to distributed protocols, which is why RMP allows this tradeoff to be made on a per packet basis. RMP demonstrates that a fault tolerant, reliable, atomic, fully distributed, totally ordered multicast protocol can actually achieve much better performance in group communication than systems that don't provide these features. For a detailed discussion of the performance of RMP, the reader is referred to [24, 23].

The biggest decision in building a reliable multicast protocol is how to guarantee the reliability and stability of messages without sacrificing throughput or latency. Latency is defined as the time between when a site has a packet to send and when it is delivered to the destination. A message is defined as going stable when the sender knows all destinations have received it. This is the point at which it no longer needs to be held for possible retransmissions. In a reliable multicast protocol, one of factors influencing throughput is the number of ACKs sent per packet, so it is important to minimize this. In order to provide guarantees of total ordering and atomic delivery in the face of failures, a reliable multicast protocol will often delay delivering a packet until after it has received one or more acknowledgments of delivery. This latency for guarantees tradeoff is fundamental to this class of protocols, which is why RMP allows this tradeoff to be made on a per packet basis.

Traditional protocols use positive acknowledgments (ACKs) from the destination to acknowledge successful receipt of a packet. While quickly providing stability of messages, this approach does not scale well to a multicast system, because each destination has to send an ACK for each packet or set of packets. The use of positive acknowledgements largely defeats the advantage of using multicast packets, because it decreases both the efficiency and the performance of the protocol. Even though positive acknowledgment messages are small, it is because they all are sent simultaneously that they can cause network congestion. In addition, having to process an ACK from each destination increases the load on the sender and decreases the performance of the protocol. One optimization is to not acknowledge every packet. In general, as the number of packets per ACK increases, the length of time for a message to go stable increases, but the lower the load is. As another approach, many systems use negative acknowledgments (NACKs). Negative acknowledgments shift the burden of error detection from the source to the destinations. Packets are stamped with sequential sequence numbers which destinations use to provide reliable delivery by detecting gaps in the sequence numbers and requesting retransmission of the packets corresponding to the gaps. Because the information that a packet has been received is never propagated back to the sender, the senders in these protocols do not ever know for certain that a destination has received a packet. Because of this, senders have to indefinitely keep a copy of each packet sent if the protocol is to be considered truly reliable. In addition, a lost packet will not be detected until another packet is received successfully, which may take a long time if the packet is the last to be sent to the ring for a while. Because of these problems, the RMP algorithm uses a combination of these two approaches. The basic algorithm is based on the ideas of the protocol originally done by Chang and Maxemchuk [11, 9].

The MBusI [8] was the original motivation for RMP. It provides a central server through which clients connect with TCP/IP streams, and an easy to use interface designed to ease the implementation of CSCW applications. It provides both total ordering of messages and reliable multicast, but has very limited scalability, since all packets must be routed through a central point, and duplicate copies sent to each destination.

The Totem protocol [22] is perhaps closest to RMP in its approach, and has reported similar throughput levels to RMP under heavy load. It also uses a rotating token ring approach, but only provides for a single ring for each broadcast domain. Totem avoids using any ACKs by only allowing the current token holder to send data. This provides high throughput under high load over a low latency network, but provides lower throughput and longer latency under low and asymmetrical loads. In addition, because it only allows a single sender to transmit at a time it will provide lower throughput over longer latency networks. To alleviate this problem they have proposed, but not implemented, gateways to link multiple broadcast domains together.

The ISIS system [28, 4] is one of the pioneering protocols in this field. It provides causal ordering and, if desired, total ordering of messages on top of a reliable multicast protocol. The reliable multicast protocol requires separate acknowledgments from each destination, which limits performance. A new system that provides causal ordering on top of IP Multicast has been implemented which is much more efficient than the old system [5], and we are comparing RMP and this new protocol.

The Psync protocol [6] is an ingenious protocol that uses piggybacked ACKs to provide causal ordering of messages and detection of dropped packets. However, both it and the similar Trans [25] and Lansis [20] protocols require that all of the members of the group regularly transmit messages. The Trans protocol and the ToTo protocol [19] implemented on top of Lansis both provide total ordering of messages. These algorithms require that at least a majority of the group members be heard from before a message can be delivered, which causes latency to increase by at least an order of magnitude. For example, for the ToTo protocol to send to a group of 8 destinations under heavy, periodic load from all sources (the best case), the latency is 23.8 ms. This increases to 114.1 ms for lightly loaded poisson sources.

The Multicast Transport Protocol (MTP) [15] is an example of an asymmetric reliable multicast protocol. One site is the communication master which grants "tokens" to group members to allow them to send data. These tokens provide both flow control and total ordering of messages. This causes over dependency on the master, which limits both reliability and performance. MTP also relies exclusively on NACKs for error recovery, which limits reliability and requires extreme amounts of buffer space.

The protocol by Crowcroft and Paliwoda [12] is one of the first protocols to propose reliable multicast over an internetwork which supports hardware multicast. The protocol provides different levels of reliability guarantees, and uses positive acknowledgments from all destinations for reliability. The paper analyzes the flooding problems that occur with simultaneous ACKs from many destinations and proposes a windowed flow control system, in some ways similar to that used in RMP, to alleviate these problems. The xAmp protocol [29] is distributed but also waits for ACKs from all destinations, and so will exhibit performance similar to the earlier versions of ISIS.

The broadcast protocol proposed by Kaashoek et. al. [17] uses a central token site to serialize messages and NACKs for retransmissions. It piggybacks ACKs onto sent messages and has the token site regularly contact silent sites in order to limit buffer space. This protocol has reported very good latency (as low as 1.3 ms for a NULL packet) because it has been implemented on top of bare hardware. However, because each message must be transmitted twice it will fundamentally achieve lower throughput than RMP – 600 KB/sec is a rough upper bound for a 1250 KB/sec Ethernet, as compared to 842 KB/sec for RMP. This will also limit the latency for larger messages; as a 8KB packet in their protocol will spend a minimum of 13.1 ms on the Ethernet, as opposed to 6.7 ms for the message and ACK of RMP.

3 Atomicity, Reliability and Ordering

Different multicast applications require many different levels of reliability and ordering guarantees in the face of transient network failures such as dropped packets. These applications also require different atomicity guarantees in the face of site failures or partitions. For example, a CSCW application may need packets to be reliably delivered at all sites, with the packets from the same source delivered in the order they were sent. This application may be able to continue, even if some sites fail away or the group partitions in two. On the other hand, a distributed database may require that all packets be delivered in the same total order at all sites, even if some of the sites partition away. RMP supports a wide range of guarantees on packets by allowing different QoS levels to be specified for packets being sent to a group, and by allowing applications to specify the minimum size of a partition that can continue to function in the face of failures.

The basic RMP QoS levels are unreliable, reliable, source ordered, and totally ordered. They are provided by differing the time at when packets are delivered and enabling or disabling the duplicate detection, NACK, and ACK policies. While throughput remains similar for the different QoS levels, higher QoS levels increase the latency of packet delivery. For example, in the common case of few dropped packets, source ordered packets have about the same latency as unordered packets, and totally ordered packets have about twice the latency of either.

The unreliable QoS is most similar to UDP traffic. An unreliable packet will be delivered 0, 1, or more times to a destination and there are no ordering guarantees on delivery. A reliable packet will be delivered 1 or more times to each destination. The source ordered QoS provides the equivalent guarantees of running a TCP socket from each source to each destination. Packets arrive exactly once at each destination in the same order as they were sent from the sender. Source order does not provide any guarantees on the ordering of packets from multiple senders in the group. Totally ordered delivery serializes all of the packets to a group, delivering all of the packets in the same order at all members of the group. Without globally synchronized clocks, it is not possible to tell which message was "really" sent first, but total ordering guarantees that some order will be imposed over all messages sent to a group, and that messages will be delivered in this order at all sites. This QoS is equivalent to running a TCP socket from each source into a central bus which serializes the packets and then sends them out through a separate TCP socket to each destination. Totally ordered packets are also causally ordered, as per Lamport's definition [21].

ISIS first defined the notion of virtual synchrony [28, 4]. Virtual synchrony often allows a distributed application to execute as if its communication was synchronous, when it is actually asynchronous. The key requirement for virtual synchrony is that all sites see the same set of messages before and after a group membership change. In other words, for a given set of packets delivered to a group, a membership change operation will partition these packets into the same two sets at all sites, and all packets in the first set will be delivered at all sites before any packets are delivered in the second set. RMP provides virtual synchrony for packets that have a QoS of at least totally ordered. This is done by implementing each membership change as a packet with a totally ordered QoS.

4 Fault Tolerance

A critical question in group protocols is what happens to delivery guarantees in the face of failures or partitions. To solve this problem, RMP offers four levels of fault-tolerant guarantees: atomic delivery within partitions, K-resilient atomic between partitions, agreed delivery between partitions, and safe delivery between partitions. The exact semantics of agreed and safe delivery are defined in [19]. All of these guarantees rely on a method of failure detection based on timeouts. If communication to one or more group members fails for an extended period of time (say 15-30 seconds), the RMP failure membership algorithm will remove them from the group. If this is due to a temporary partition, they can later rejoin the group, but as new members.

Because a member can not join back in to a ring as an old member once it has been removed, it is not possible for a group to partition into two halves and then rejoin. This is the key to atomic delivery within partitions. RMP guarantees that for totally ordered packets, if any member in a partition delivers a packet, all of the other members of that partition will deliver that packet if they were in the group membership view when the packet was sent and if they remain in the group for a sufficient period of time. Since no totally ordered packet will be discarded until it has become stable within a partition, and no packet can become stable within a partition until it has been received (but not necessarily delivered) by all of the members of the partition, if any site delivers a packet, all of the other sites in the partition will receive it before it is discarded by that site. Once a site has a packet, the only way it will not deliver it is if it crashes, upon which case it will be detected and removed from the group.

This level of atomicity does not provide any guarantees about delivery or ordering of packets between partitions. K-resilient atomicity between partitions is the first level of guarantee that addresses this. K is the minimum number of sites that must fail or partition away from a group over a short period of time, in order to violate atomicity guarantees. This is provided by having each member M verify that at least K other members have received a packet before M can deliver it. In this case, each partition will always have at least one member which has received all of the packets that have been delivered at any site.

The next level of atomicity is called agreed ordering, or majority resilience. Agreed ordering guarantees that no matter how many partitions or failures occur, any members of a group that deliver any two messages will agree on the same ordering of the messages. This level guarantees total ordering across partitions, but not atomicity. This level of atomicity is achieved by not allowing minimum partitions to continue and by making sure the majority of the members of a group have a message before any member delivers it. Both of these tests require a possibly conservative calculation of how many members are in the group, which we call MaxN. MaxN is equal to the maximum number of members that are in any membership view for any packet which is not yet stable.

The final level of fault tolerance is safe delivery, also called total resilient delivery. This level requires that a packet be stable before it can be delivered. This occurs after the token has been passed once around the ring after a packet was received. At this point, the member knows that all other members have received it, but they may not all deliver it. It is still possible that one or more of the sites could fail before delivering the packet. This is the highest level of atomicity that any system such as RMP can reasonably provide.

Ordering guarantees between packets of different QoS levels are determined by the lowest QoS of the packets in question. For example, for a set of packets S1 with source ordered QoS and a set of packets S2 with totally ordered QoS, the best guarantee that is provided over the union of the two sets is source ordering.

5 Group Communication

The two main options in current communication addressing are explicit and implicit addressing. RMP supports both of these addressing models, and supports both the peer group model of communication and the the client-server model of communication. Protocols such as TCP and UDP require explicit naming of the destinations of communication, while systems such as Grapevine [26] and the MessageBus [8] allow implicit naming through a publisher/subscriber model of communication. With implicit naming, RMP processes join or "subscribe" to a group by specifying the name of a group to join, and other processes "publish" or send messages to

```
if (A < MIN_PACKET && A < W)
  then Delay sending packet until an ACK is received
// Send up to 1/2 of the window at a time
S = min(P, W/2);
// Send at least MIN_PACKET bytes
S = max(S, MIN_PACKET);
// Can only send up to A bytes
S = min(S, A);
// Reduce effect of lost packets
S = min(S, MAX_PACKET);
```

Figure 1: Flow control algorithm to determine size of packet to be sent

this group by using the group name or a membership view ID associated with the name. When this model is used, messages sent to the group name are delivered automatically to all RMP processes, if any, that are members of that group, so no explicit knowledge of the membership of a group is needed. As explained above, RMP does this by mapping group names into multicast address, port, TTL tuples that are used to send to other members of the group.

Instead of specifying a group by its name, RMP processes may explicitly name another RMP process (specified by an IP address and a UDP port) that is a member of the group and request that this member forward packets on behalf of it. In addition to allowing members to join a group based on a process ID, rather than a name, this can also be used by a non-member or non-multicast capable member to send packets to a group. When coupled with the notification of members of the current membership of the group at whenever the membership changes, this allows processes to exert explicit control over group naming and membership when desired.

6 Multi-RPC Delivery

The client-server model of communication has become widely accepted as a powerful way of providing services to users. While RMP could support this model simply by having all clients and servers join a group, this is often inefficient and will limit the scalability of client/server groups. As an alternative, RMP provides facilities for RMP processes that are not members of a group to use a multi-RPC algorithm to send data to a ring and to receive acknowledgments of successful delivery and/or responses from a member of the group. This is a powerful feature, for it allows multiple servers to exist in a group, and all of them can get messages from clients. These messages can be automatically acknowledged, or a single member of the group can be selected to handle the request and reply to it. These Non Member Data packets can be delivered with all of the QoS levels of a Data packet sent from a group member.

7 Flow and Congestion Control

Flow and congestion control policies for reliable multicast protocols are an open problem. Because reliable multicast protocols primarily use NACKs for error detection, there is no existing explicit feedback path with which destinations can signal losses or low buffer space to the senders. In addition, the throughput for a multicast group should be divided up between the members of the group who are trying to send, but the policy for this division is usually dynamic and not known in advance. Because of this, the flow and congestion control policies used by

RMP are designed to be orthogonal to the rest of the protocol. Flow and congestion control policies can be inserted easily into the protocol, and different policies can be used in different environments. As the default, we propose a modified sliding window protocol based on the Van Jacobson algorithms used in TCP [18].

One problem that RMP faces with flow and congestion control is that the rotating token site introduces a higher overhead per acknowledgment than traditional protocols such as TCP. This is compounded by the protocol being more complicated than TCP and thus requiring more processing per packet. To solve this problem, RMP uses larger packet sizes than does TCP. In an error free environment, having the IP or IP Multicast layer do the fragmentation and re-assembly is more efficient than having RMP do it. If errors occur, the window size quickly drops to a single minimum size packet. The algorithm to determine the size of the packet to be sent out (S), given the current window size (W), the available space in the window (A), and the offered packet size (P), is shown in Figure 1. The critical step in this algorithm is that up to half of the available window is sent at a time until the maximum packet size has been reached. This trades off a small amount of network utilization in the case of errors for typically higher efficiency of handling packets and higher throughput.

8 Conclusions

Data replication will emerge as an important consideration for efficiency as Internet bandwidth and population increase, as the price-performance of processors decrease and storage capacities grow. Distributed databases and groupware applications will depend on robust, low-level protocols for reliable delivery to maintain coherence between copies of data on large numbers of host machines.

In this paper we have described the basic questions and applications of reliable multicasting in the conext of the development and analysis of a reliable multicast protocol. RMP is a fully distributed reliable multicast protocol with selectable ordering, atomicity, and fault tolerant guarantees that can be used to implement distributed applications. We have shown that RMP provides these features with very high performance [23]. To our knowledge, RMP provides better performance for totally ordered delivery of packets to 2 or more destinations on an Ethernet than any other protocol. Much work has gone into providing reliable multicast services with lower ordering guarantees because it was believed that the performance of a totally ordered multicast protocol was inherently low. Our experience with RMP suggests that this is not the case, and that an efficient reliable multicast service can provide total ordering of messages for only a small latency penalty. Because of its use of multiple groups, an optional client-server architecture, its fully distributed nature, and its flow and congestion control algorithms, we expect RMP to scale gracefully and efficiently to large groups spread over a large internetwork. Finally, RMP can take technologies such as wireless and satellite communications that support broadcast and multicast transmission. Initial results confirm these hypotheses and we plan to continue our analysis in the near future.

References

[1] T. Bell J. Adam and S. Lowe. Technology 1996: Communications. *IEEE Spectrum*, 33(1):30–41, January 1996.

[2] R. D. Barbara Alonso and H. Garcia-Molina. Data caching issues in an information retrieval system. *IEEE Transactions on Information Systems*, 1989.

[3] K. Scott Barry, M. and S. Weismuller. A distributed computing model for telemetry data processing. In *Proceedings of 9th NASA/GSFC Conference on Space Applications of Artificial Intelligence*, May 1994.

[4] K. Birman. The Process Group Approach to Reliable Dsitributed Computing. *Communications of the ACM*, 36(12):37–53, December 1993.

[5] K. Birman and T. Clark. Performance of the Isis Distributed Computing Toolkit. Technical Report TR-94-1432, Cornell University, December 1994.

[6] L. L. Peterson N. C. Buchholz and R.D. Schlichting. Preserving and using context information in interprocess communication. *ACM Transactions on Computer Systems*, 7(3):217–246, August 1989.

[7] J. Callahan and T. Montgomery. An approach to verification and validation of a reliable multicast protocol. In *Proceedings of the ACM Internation Symposium on Software Testing and Analysis (ISSTA)*, January 1996.

[8] A. Carroll. *ConversationBuilder: A Collaborative Erector Set.* PhD thesis, Department of Computer Science University of Illinois, 1993.

[9] J. M. Chang and N. F. Maxemchuk. Reliable Broadcast Protocols. *ACM Transactions on Computer Ssystems*, 2(3):251–273, August 1984.

[10] J.M. Chang. Simplifying distributed database systems design by using a broadcast network. In *SIGMOD '84*, pages 223–233, June 1984.

[11] J.M. Chang and N.F. Maxemchuk. A broadcast protocol for broadcast networks. In *GLOBCOM '83*, December 1983.

[12] J. Crowcroft and K. Paliwoda. A multicast transport protocol. In *ACM SIGCOMM '88*, pages 247–256, 1988.

[13] M. Schwartz Danzing, P. and R. Hall. A case for caching file objects inside internetworks. In *Proceedings of the ACM SIGCOMM '93*, pages 239–248, September 1993.

[14] S. Deering. Host Extensiosn for IP Multicasting. Technical Report RFC-1112, IETF, August 1989.

[15] S. Armstrong A. Freier and K. Marzullo. Multicast Transport Protocol. Technical Report RFC-1301, IETF, February 1992.

[16] Object Management Group. The common object request broker: Architecture and specification. Technical Report 91.12.1, Object Management Group, 1991.

[17] M. F. Kaashoek A. S. Tanenbaum S. F. Hummel and H. E. Bal. An efficient reliable broadcast protocol. *Operating Systems Review*, 23(4):5–19, October 1989.

[18] V. Jacobson. Congestion Avoidance and Control. In *SIGCOMM Proceedings*, pages 314–328. ACM, 1988.

[19] D. Dolev S. Kramer and D. Malki. Early delivery totally ordered multicast in asynchronous environments. In *23rd Annual International Symposium on Fault-Tolerant Computing (FTCS)*, pages 544–553, June 1993.

[20] Y. Amir D. Dolev S. Kramer and D. Malki. Transis: A Communication Sub-system for High Availability. Technical Report CS9113, Hebrew University of Jerusalem, November 1991.

[21] L. Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. *Communications of the ACM*, 21(7):558–565, July 1978.

[22] D. Agarwal P. Melliar-Smith and L. Moser. Totem: A Protocol for Messaging Ordering in a Wide-Area Network. In *First ISMM International Conference on Computer Communications and Networks*, pages 1–5, June 1992.

[23] B. Whetten T. Montgomery and S. Kaplan. A High Performance Totally Ordered Multicast Protocol. In *Theory and Practice in Distributed Systems*, number 938 in LCNS. Spring Verlag, 1994.

[24] T. Montgomery. Design, Implementation, and Verification of the Reliable Multicast Protocol. Master's thesis, West Virginia University, December 1994.

[25] P. M. Meillar-Smith L. E. Moser and V. Agrawala. Broadcast protocols for distributed systems. *IEEE Transactions on Distributed Systems*, 1(1):17–25, January 1990.

[26] A. Birrell R. Levin R. Needham and M. Schroeder. Grapevine: An exercise in distributed computing. *Communications of the ACM*, 25(4):260–274, April 1982.

[27] K. Ravindran and X. T. Lin. Structural complexity and execution efficiency of distributed application protocols. In *Proceedings ACM SIGCOMM '93*, pages 160–169, September 1993.

[28] K. Birman A. Schiper and P. Stephenson. Lightweight Causal and Atomic Group Multicast. *ACM Transactions on Computer Systems*, 9(3):272–314, August 1991.

[29] Verissmo. xamp: A multi-primitive group communications service. In *Proceedings of the 11th Symposium on Reliable Distributed Computing*, 1992.

[30] Y. Wu. The specification-based validation of rmp. Master's thesis, West Virginia University, December 1995.