

Reliable Dissemination for Large-Scale Wide-Area Information Systems

Rajendra Yavatkar James Griffioen
Department of Computer Science
University of Kentucky

Abstract

This paper describes a reliable multicast protocol (TMTP) that can be built at the transport layer without modification to the network-level routers. The protocol combines the best of sender and receiver initiated protocols to distribute protocol processing and localize retransmission. Experimental results obtained across Internet MBONE sites in the USA and Europe are presented.

1 Introduction

Over the past few years the Internet has emerged as a communication utility as ubiquitous as the phone service. This change has had a dramatic affect on the way people communicate and share information. As a result, it has spawned widespread development efforts focusing on wide-area distributed information systems. Wide-area information systems include applications for information exchange between users, dissemination of information to users, or even communication of data amongst parallel processes. These wide-area information systems encompass such application domains as electronic publishing, news/newsgroup services, electronic mailing lists, interactive collaborative applications, distributed games, distributed parallel programming systems, distributed discrete-event simulations, and digital libraries. Until recently, however, lack of efficient support for wide-area group communication made it impractical to design and implement such applications on the scale of a wide area network such as the Internet. In recent years, the deployment of the MBONE and the widespread availability of IP (UDP) multicast capability [2, 1] has renewed interest in designing wide-area information systems around a group communication paradigm.

The goal of our research in group communication is to create a flexible and scalable group communication framework based on IP/UDP multicast. We are interested in the design and implementation of efficient, scalable transport layer protocols for such a framework. In particular, we have identified three distinct styles of group communication, namely, *dis-*

semination, concast, and conversation that frequently arise in many application domains.

Dissemination involves $1 \times N$ communication in which a single sender multicasts (or disseminates) information to members of a group. Examples of such communication include electronic publishing, distribution of state information (e.g., routing table updates), and whiteboard style presentations [6]. Typically, such applications require reliable delivery of messages to all the receivers, but do not require guarantees on atomic delivery or causal ordering of communication because the group communication does not involve multiple senders.

Concast-style communication involves $N \times 1$ communication in which many group members send several messages to a single recipient. For example, distributed monitoring applications gather information at points throughout the system and periodically report the gathered information to a single site for processing/decision making. Another example of such communication arises in "report-in/report-back" applications (such as distributed, interactive games) when a group member issues a request to multiple receivers who process the request and report the answer back to the requesting member. Again, even though multiple senders are involved, constraints such as atomicity and causality that are of concern in general group communication do not arise here, but reliable delivery is important and care should be taken not to overload the network or the receiver when replies of several group members arrive at the same time.

Conversation-style is the most general form of group communication and involves $M \times N$ communication in which many participants may both send and receive messages. Examples of applications involving such communication include distributed group editors and certain collaborative systems [3, 5]. In this case, reliable delivery subject to constraints such as causality and atomicity is often important. Protocols such as Psynch and those developed in conjunction with the ISIS project support this style of communication

and are not necessarily optimized to yield good performance for the other two styles of group communication.

Our recent efforts have concentrated on the design and implementation of a transport protocol based on IP multicast that is explicitly designed to achieve large-scale, reliable *dissemination* and *concast* communication across a wide area network. The protocol called *TMTP* (Tree-based Multicast Transport Protocol) takes advantage of IP multicast for efficient multicast routing and packet delivery but augments it with scalable methods for flow control and error recovery to achieve reliable dissemination and concast.

This paper provides a brief introduction to TMTP and its novel features. We also present experimental results across the Internet MBONE. Additional details can be found in [10].

2 TMTP Dissemination Model

Under the TMTP dissemination model, a single sender multicasts a stream of information to a *dissemination group*. A *dissemination group* consists of processes scattered throughout the Internet, all interested in receiving the same data feed. A session directory service (similar to the session directory *sd* from LBL [4]) advertises all active dissemination groups.

Before a transmitting process can begin to send its stream of information, the process must create a dissemination group that initially consists of no members. Once the dissemination group has been formed, interested processes can dynamically join the group to receive the data feed. The dissemination protocol does not provide any abstraction or mechanism to insure that all receivers are present and listening before transmission begins and, if necessary, an application must use a coordination mechanism outside TMTP before beginning group communication. Thus, TMTP allows for dynamic membership scenarios that might arise in applications such as electronic publishing where receiver processes usually join a data feed already in progress and/or leave a data feed prior to its termination.

Once the communication begins, TMTP achieves reliable and large-scale dissemination using the following techniques:

1. TMTP takes advantage of IP multicast for fast packet routing and delivery.
2. TMTP uses an *expanding ring search* to dynamically organize the dissemination group members into a *hierarchical control tree* as members join and leave a group.
3. TMTP achieves scalable reliable dissemination via the hierarchical control tree used for flow and

error control. The control tree takes the flow and error control duties normally placed at the sender and distributes them across several nodes. This distribution of control also allows error recovery to proceed independently and concurrently in different portions of the network.

4. Error recovery is primarily driven by receivers who use a combination of *restricted negative acknowledgments with nack suppression* and periodic positive acknowledgments. In addition, the tree structure is exploited to restrict the scope of retransmissions to the region where packet loss occurs; thereby insulating the rest of the network from additional traffic.

The control tree structure is also used in the case of concast communication. In this case, intermediate nodes combine many small messages (data and control) into a large message and forward them across the branches of the tree for scalable delivery.

We have completed a user-level implementation of TMTP based on IP/UDP multicast and have used it for a systematic performance evaluation of reliable dissemination using the current Internet Mbone. Our experiments have involved as many as thirty group members located at several sites in the US and Europe. The results are impressive; TMTP meets our objective of scalability by significantly reducing the sender's processing load, the total number of retransmissions that occur, and the end-to-end latency (a factor of four improvement) as the number of receivers is increased.

3 TMTP's Group Management

For the purposes of flow and error control, TMTP organizes group participants into a hierarchy of subnets or *domains*. Typically, all the group members in the same subnet belong to a domain and a single *domain manger* acts as a representative on behalf of the domain for that particular group. The domain manager is responsible for recovering from errors and handling local retransmissions if one or more of its children do not receive some packets.

The domain managers are organized into a *control tree*. The sender in a dissemination group serves as the root of the tree and has at most K domain managers as children. Each domain manager will accept at most K other domain managers as children, resulting in a tree with maximum degree K . The degree of the tree (K) limits the processing load on the sender and the internal nodes of the control tree. Furthermore, the protocol overhead grows slowly, proportional to $\text{Log}_K(\text{Number_Of_Receivers})$.

The control tree is self-organizing and does not rely on any centralized coordinator, being built dynamically as members join and leave the group. A new domain manager attaches to the control tree by discovering the closest node in the tree using an *expanded ring* search. Similarly, a domain manager with no remaining local clients may remove itself from the tree by asking its children to re-attach themselves. Note that *the control tree is built solely at the transport layer and thus does not require any explicit support from, or modification to, the IP multicast infrastructure inside the routers.*

4 The Transmission Protocol

Packet transmission in TMTP proceeds as follows. When a sender wishes to send data, TMTP uses IP multicast to transmit packets to the entire group. The transmission rate is controlled using a sliding window based protocol described below. The control tree ensures reliable delivery to each member. Each node of the control tree (including the root) is only responsible for handling the errors that arise in its immediate K children. Both receivers and domain managers only send periodic, positive acknowledgments to their immediate parent, never to the distribution source, thereby avoid packet implosion at the source. When a child detects a missing packet, the child multicasts a NACK in combination with *nack suppression*. To limit the scope of the multicast NACK and the ensuing multicast retransmission, TMTP uses the *Time-To-Live* (TTL) field to restrict the transmission radius of the message to its clients and domain manager children. As a result, error recovery is completely localized. Thus, a dissemination application such as a world-wide IETF conference would organize each geographic domain (e.g., the receivers in California vs. all the receivers in Australia) into separate subtrees so that error recovery in a region can proceed independently without causing additional traffic in other regions. TMTP's hierarchical structure also reduces the end-to-end delay because the retransmission requests need not propagate all the way back to the original sender. Because packets are retransmitted locally, retransmitted packets have a short propagation delay and are received quickly by the affected receivers. Nacks further speed the recovery process, increasing the probability of complete and timely positive acknowledgments being sent back up the tree.

TMTP's primary means of flow control consists of a window-based approach used for both dissemination from the sender and retransmission from domain managers. Within a window, senders transmit at a fixed rate. TMTP uses two different timers to control the

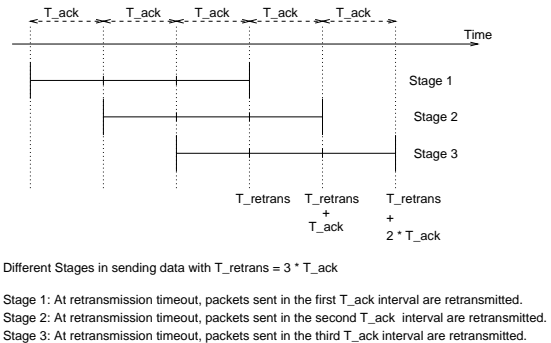


Figure 1: TMTP's window-based flow control.

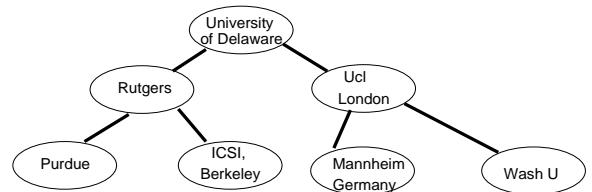


Figure 2: The hierarchical control tree used in our experiments.

window size and the rate at which the window advances. $T_{retrans}$ defines a timeout period that begins when the first packet in a window is sent. A second timer, T_{ack} , defines the periodic interval at which each receiver is expected to unicast a positive ACK to its parent. The sender starts a timer and begins transmitting data (at a fixed rate). Although the sender should see a positive ACK at time T_{ack} , the sender does not require one until time $T_{retrans}$. Figure 1 illustrates the windowing algorithm graphically.

There are three reasons for using multiple T_{ack} intervals during a retransmission timeout interval ($T_{retrans}$). First, by requiring more than one positive ACK during the retransmission interval, TMTP protects itself from spurious retransmissions arising from lost ACKs. Second, a larger retransmission interval gives receivers sufficient time to recover missing packets using receiver-initiated recovery (NACKs) when only one (or a few) packets in a window are lost. This avoids unnecessary multicast retransmissions of a window full of data. Third, multiple T_{ack} intervals during the timeout interval provide sufficient opportunity for a domain manager to recover from transient network load in its part of the subtree without unnecessarily applying backpressure to the sender.

5 Experimental Results

Figure 2 illustrates the environment in which the experiments were run. Our tests involved seven ge-

ographically distinct sites connected by the Internet Mbone. All of our experiments were conducted using standard IP multicast across the Internet Mbone and thus experienced real Internet delays, congestion, and packet loss.

As a point of comparison, we implemented the standard sender-initiated reliable multicast transport protocol both with and without window-base flow control (called *WIN_BASEP* and *BURST_BASEP* respectively). Under both protocols, the sender maintains state information for all receivers, expects positive ACKs from each receiver, and uses timeouts and multicast retransmissions to recover from missing acknowledgments. The protocols illustrate the performance bottlenecks related to processor load and end-to-end latency. All three protocols used the same packet size (1K bytes). TMTP and WIN_BASEP used a window size of 5 and a transmission rate of 10 packets per second.

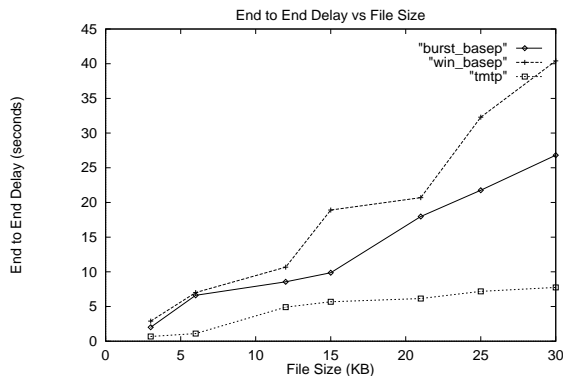


Figure 3: The end-to-end delay required to delivery a file to 30 group members spanning 7 sites.

Figures 3 and 4 show the end-to-end delay of each algorithm (i.e., the time required to reliably transmit a complete file to multiple receivers). Figure 3 shows how well each algorithm scales as the file size increases, while Figure 4 show how well each algorithm scales as the number of receivers increases. The balanced nature of our control tree meant the event processing load was spread equally among the sender and all domain managers. For a given file size, the number of events (ACK/NACK processing, timeouts, and retransmissions) processed by the sender remained constant, regardless of the number of receivers; whereas the two BASEP protocols experienced a linear increase as new receivers were added. Most timeouts/retransmissions occurred as a result of dropped ACKs. TMTP substantially reduced the

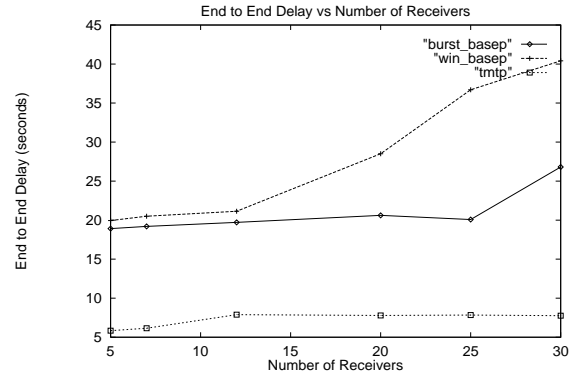


Figure 4: The end-to-end delay required to deliver a 30K file to multiple recipients.

ACK implosion and only had 6 local retransmissions totaled across all domain managers (four occurring concurrently) as opposed to 9 global retransmission for BURST_BASEP (out of thirty 1K messages).

6 Related Work

Earlier multicast protocols used conventional flow and error control mechanisms based on a *sender-initiated* approach in which the sender disseminates packets and uses either a *Go-Back-N* or a *selective repeat* mechanism for error recovery. Under this approach, the sender must maintain a large amount of state information and may experience *packet implosion* from a large number of ACKs leading to an overall decrease in throughput [8].

Receiver-initiated methods [9, 6] shift the burden of reliable delivery to the receivers. Each receiver maintains state information and explicitly requests retransmission of lost packets via NACKs. However, the recovery time may be arbitrarily large as it depends solely on the timeouts at the receivers which may not detect errors such as lost packets at the end of a packet train [9]. Also, the sender does not receive positive ACKs required to efficiently reclaim buffer space. TMTP incorporates the best of the sender and receiver initiated approaches into a single model.

Recently Paul et. al. [7] have proposed and are examining three multicast alternatives with features similar to those of TMTP. In contrast to these protocols, TMTP uses a multi-level hierarchical control tree and a dynamic group management protocol, as opposed to a static two-level hierarchy, to evenly distribute the protocol processing load and allow finer grained independent and concurrent error recovery. TMTP targets a best-effort multicast system such as IP multicast rather than an ATM-like network with

allocated resources. Furthermore, TMTP imposes no additional load on network-level routers and requires no modification to the network-level routers, but yet incorporates both local retransmissions and combined acknowledgments. TMTP also employs receiver-initiated techniques and a unique flow control mechanism that can tolerate transient congestion and lost acknowledgments.

References

- [1] S. Casner and S. Deering. First IETF Internet Audio-cast. *ACM Computer Communication Review*, 22(3):92–97, July 1992.
- [2] Stephen E. Deering and David R. Cheriton. Multicast routing in datagram internetworks and extended lans. *ACM Transactions on Computer Systems*, 8(2):85–110, May 1990.
- [3] Prasun Dewan. A Guide to Suite: Version 1.0. Technical Report SERC-TR-60-P, Software Engineering Research Center, Purdue University, West Lafayette, IN, February 1990.
- [4] Van Jacobson. *SD: Session Directory*. Lawrence Berkeley Laboratory, March 1993.
- [5] Amit Mathur and Atul Prakash. Protocols for integrated audio and shared windows in collaborative systems. In *Proceedings of ACM Multimedia '94*, October 1994.
- [6] Steven McCanne. A Distributed Whiteboard for Network Conferencing. Technical report, Real Time Systems Group, Lawrence Berkeley Laboratory, Berkeley, CA, September 1992. unpublished report.
- [7] S. Paul, K. Sabnani, and D. Kristol. Multicast Transport Protocols for High Speed Networks. In *IEEE Int. Conf. on Network Protocols*, 1994 Oct.
- [8] Sridhar Pingali, Don Towsley, and James F. Kurose. A comparison of sender-initiated and receiver-initiated reliable multicast protocols. In *Proceedings of ACM SIGMETRICS '94*, volume 14, pages 221–230, 1994.
- [9] S. Ramakrishnan and B.N. Jain. A Negative Acknowledgement Protocol with Periodic Polling Protocol for Multicast over Lans. In *Proceedings of IEEE INFO-COMM '87*, pages 502–511, March-April 1987.
- [10] R. Yavatkar and J. Griffioen. TMTP: A Scalable Transport Protocol for Interactive Multipoint Applications. Technical report, Department of Computer Science, University Of Kentucky, April 1995. submitted for publication.