Overview of the InterGroup Protocols*

K. Berket[1], D. A. Agarwal[1], P. M. Melliar-Smith[2], and L. E. Moser[2]

[1] Ernest Orlando Lawrence Berkeley National Laboratory
1 Cyclotron Rd, MS 50B-2239
Berkeley, CA 94720
{KBerket, DAAgarwal}@lbl.gov
[2] Department of Electrical and Computer Engineering
University of California, Santa Barbara
Santa Barbara, CA 93106
{pmms, moser}@ece.ucsb.edu

Abstract. Existing reliable ordered group communication protocols have been developed for local-area networks and do not, in general, scale well to large numbers of nodes and wide-area networks. The InterGroup suite of protocols is a scalable group communication system that introduces a novel approach to handling group membership, and supports a receiver-oriented selection of service. The protocols are intended for a wide-area network, with a large number of nodes, that has highly variable delays and a high message loss rate, such as the Internet. The levels of the message delivery service range from unreliable unordered to reliable group timestamp ordered.

1 Introduction

Distributed applications often need to maintain consistency of replicated information and coordinate the activities of many processes. Collaborative applications and distributed computations are both examples of these types of applications. With the advent of grids [8], distributed computations will be spread across multiple computer centers requiring efficient mechanisms for coordination between the processes. Collaborations are by their very nature distributed and built in an incremental, ad hoc manner. Group communication provides a very natural mechanism for supporting these types of applications and allowing them to use a peer-to-peer architecture rather than a server-based architecture.

The MBone videoconferencing tools (vic, vat, and rat), the session directory tool (sdr) and the whiteboard tool (wb)[1] are excellent examples of the peer-to-peer model. These tools are designed to use multicast protocols to send data, which allows groups to form and communicate without coordination with

* This work was supported by the Director, Office of Science. Office of Advanced Scientific Computing Research. Mathematical, Information, and Computational Sciences Division, U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

[1] More information on all of these tools and their binaries can be found at http://www-itg.lbl.gov/mbone.

a server. This peer-to-peer model inherently makes the tools easier to design and to operate for groups of two and groups of hundreds. Because there are no servers, groups can be formed in an ad hoc manner with no setup. Scheduling using a centralized authority is sometimes used in these systems.

There are many applications that can benefit from the use of a peer-to-peer group communication capability. Instant messaging systems, shared remote visualization, shared virtual reality and collaborative remote control of instruments are just a few examples. Most of these applications currently use a central server to collect messages and forward them to the participants. A peer-to-peer group communication service that provides group membership services, reliable ordered message delivery, and progress in the presence of process and network faults can be used to allow the participants to talk directly to each other.

Although group communication systems can provide these services, the protocols have historically been limited in their scalability. The InterGroup protocol suite, described in this paper, is a group communication system that tackles scalability by taking a novel approach to providing these services. Our solution includes redefining the meaning of group membership, allowing voluntary membership changes, adding a receiver-oriented selection of delivery services (which permits heterogeneity of the receiver set), and providing a scalable reliability service. The InterGroup protocols are designed specifically with the intention of scaling to the Internet and to large numbers of participants.

2 Related Work

Group communication systems provide reliable group ordered message delivery and membership services that allow the system to make progress in the presence of process and network faults. The main concerns in scaling these protocols are flow control, congestion avoidance, the reliable multicast service, and the membership service. The membership service needs to support a reliable group-ordered delivery service and a form of virtual synchrony [5, 12]. Because of this, it is essential for the membership service to be based on protocols that reach group-wide consistent decisions.

Traditionally, group communication systems, such as the Totem Single Ring Protocol (SRP) [3], Transis [2], Isis [5], and RMP [18], were designed for the local-area network environment, where network latencies and losses are minimal, and there is a small number of processes. Because of this, scalability concerns were not vital in the design of these systems.

Recently, group communication systems have made advances beyond the single local-area network environment, and research into scaling the membership services has intensified. The Totem Multiple Ring Protocol (MRP) [1] uses a hierarchy of rings interconnected by gateways. Each ring is an instantiation of the Totem SRP and the gateways provide coordination and message forwarding between the rings.

To reduce the costs encountered in applications requiring the use of a large number of process groups, dynamic light-weight groups [9] were introduced. The idea is to map this large number of application process groups to a smaller number of protocol process groups. This concept is implemented by the process group interface of Totem [13], which provides a static mapping of the application groups to one protocol group, and by the gateways of the Totem MRP, which filters the forwarding of messages based on application groups.

Another trend in group communication systems has been the breaking up of the system into building blocks. Horus [17] (and its follow-on Ensemble[16]) introduced the building block approach to group communication systems. This approach allows more flexibility in the delivery services provided to the application. It also breaks away from a monolithic approach to design, allowing a better understanding of the interactions within a group communication system.

Moshe[10] is a group membership service (building block) for use by group communication systems in the wide-area environment. It provides an optimistic algorithm, for reaching a group-wide consistent decision regarding the membership, that usually finishes in one round. This is achieved by separating the membership service from the fault detection mechanisms in such a way that most membership changes can be handled as voluntary. Moshe also separates the membership service from the virtual synchrony service. For full virtual synchrony, an additional layer built on top of Moshe is necessary.

Scalable Reliable Multicast (SRM) [7] is a protocol that was designed specifically to scale to the Internet. It is not a full group communication system; it only provides the mechanisms for recovering messages in a scalable manner. It achieves this by separating the reliability mechanisms from the loss detection mechanisms, leaving those up to a higher layer. The SRM protocol exchanges *session messages* between group members to update the control information at each individual process. The original SRM protocol uses a group-wide multicast for all of its communication, which limits its scalability. One proposed solution to this problem is the organization of the group members in a self-configuring hierarchy [15].

3 The Architecture

The InterGroup protocols are designed using a building block approach. We divided the protocols into four separate modules based on functionality. The modules are control hierarchy, reliable multicast, message delivery, and process group membership.

The control hierarchy provides a scalable mechanism for the exchange of control information between sites in the system. Each site has a control process that collects and disseminates the control information for all of the processes at that site. The control hierarchy also provides mechanisms for determining message stability, and providing group-wide consistency of information to the group members. The essential components of the control hierarchy are explained in Section 4.

Reliable multicast provides mechanisms to detect missing messages, request the retransmission of messages, retransmit messages, and detect whether a message can be recovered. The detection of missing messages in InterGroup is through detection of gaps in the sequence numbers of messages from the same source. When a node detects a missing message, it requests the retransmission of the message.

Message delivery entails the ordering and delivery of messages to the application based on the delivery service chosen by the application for a process group. The essential components of message ordering and delivery are explained in Section 5.

The process group membership protocols run at each process and track the changes in the group membership. They are affected by the delivery service chosen by the application and the sending characteristics of the process. The essential components of process group membership are explained in Section 6.

4 Control Information

There are many types of control information that need to be gathered by the InterGroup protocols. The reliable multicast protocols gather information about the latency between processes (based on algorithms introduced in SRM). The buffer management protocols gather information from the reliability service of each process so messages are held in the buffers only until they have been delivered to all of the processes in the group. The membership protocols collect information from all of the processes in the group in order to reach a consistent decision regarding the group-wide logical time at which a membership change occurs, called a *cut*.[2]

This control information must be obtained from all of the processes in the group or from all of the processes in the system. To make this operation more scalable, we gather and disseminate the control information in a hierarchical manner. The structure of our hierarchy is based on the work proposed for scaling the control information exchange in SRM [15]. The logical structure of this hierarchy attempts to mimic the underlying network topology by considering the latencies between control processes. This structure improves the efficiency of the control communication.

The control processes are organized in multicast trees, with the roots referred to as *coordinators*, and the leaves referred to as *children* (Fig. 1). Each child is associated with at least one coordinator. The *local group* of a coordinator is composed of the children of that coordinator (including the coordinator itself). The *coordinator group* consists of all the coordinators. Each control process limits its communication of control information to its local group; the coordinators also communicate with the coordinator group. The frequency of control messages is regulated using a simplified version of the congestion control algorithm used by RTCP [14].

The hierarchy dynamically reorganizes based on changes in the system. A self-determination protocol is executed periodically to determine whether a control

[2] This is required for virtual synchrony.

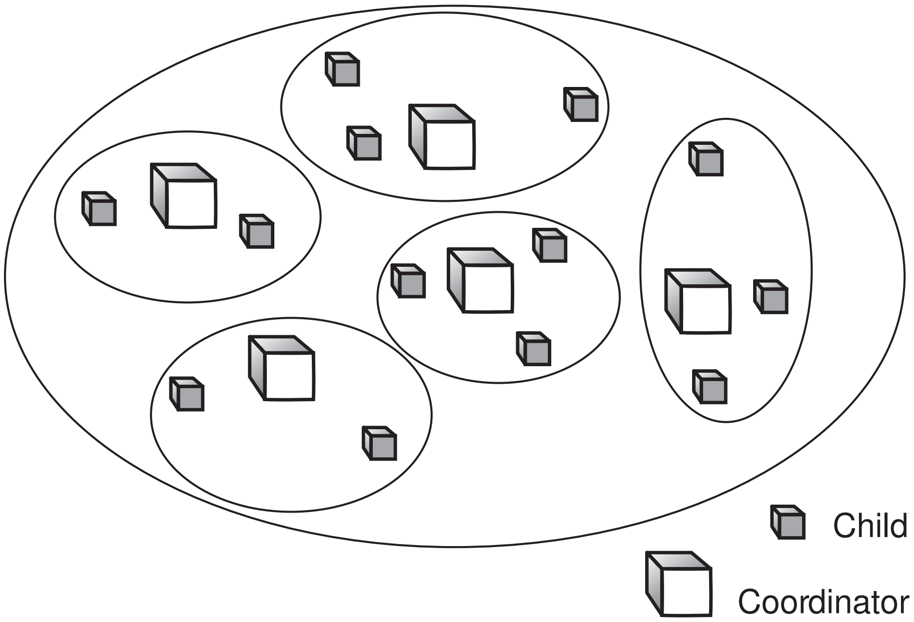

Fig. 1. The control hierarchy.

process in the hierarchy should change states (child to coordinator or vice versa) in response to changes in the system. The determination of a state change is a local decision made at each control process based on control information gathered before the self-determination protocol is executed, and based on a predefined set of rules adapted from [15].

A control process, upon startup, checks to see how many coordinators are present in the hierarchy, by checking the messages sent in the coordinator group.[3] If the number of coordinators is less than or equal to the expected average coordinator group size or the control process does not receive a control message from the coordinator group within a given time, the control process starts up as a coordinator. Otherwise, the control process starts up as a child.

A control process, starting up as a coordinator immediately starts sending control messages. A control process, starting up as a child, needs to find a coordinator that will accept it into its local group. This step is accomplished by using an expanding ring search. When it finds a coordinator, the control process starts sending messages to the coordinator's local group and becomes a child. If the expanding ring search doesn't provide a coordinator within a given time, the control process starts up as a coordinator.

The detection of control process faults is accomplished via a fault-detection algorithm that runs periodically. If the information from a control process has not been updated recently, the control process is removed from the data structures

[3] Only one message is necessary to make this determination.

and thus removed from this control process's view of the membership of the hierarchy. When a fault regarding the coordinator for this control process is detected, the control process uses the information from the self-determination protocol to decide whether it should find another coordinator and remain a child, or whether it should become a coordinator.

Global control traffic from all control processes is gathered using the hierarchy. The control information is aggregated as it is gathered through the InterGroup control hierarchy, thus controlling the control information message size. Otherwise, the size of control messages would grow proportionally with the number of control processes.

For a detailed description of the control hierarchy and a full state machine, see [4].

5 Delivery Services

The InterGroup system provides the following delivery services within a process group:

1. **Unreliable unordered**. Messages received from the process group are delivered directly to the application. Some messages might never be received, and multiple copies of the same message might be received. There is no guarantee regarding the order in which messages are received. IP Multicast provides this functionality.
2. **Reliable source ordered**. All messages from a particular source will be received by the application (unless a process fault occurs), and they will be delivered in sequence number order. This service is well-suited to applications such as multicast file transfers and many other applications currently using TCP/IP [6].
3. **Reliable group timestamp ordered**. Messages are received by the application in timestamp order over the entire process group. The membership service ensures the consistency of received messages at group members during membership changes. This service is closest to the idea of agreed messages in group communication systems [3].

Each process determines the delivery service it desires for a process group at the receiving end. To achieve this receiver-oriented delivery service, we need (1) to order messages independently at each receiver without restricting the delivery service choices of the other receivers, and (2) to separate the reliability service from the ordering service.

We make application messages "born-ordered" [12] to achieve the first goal. Each message is ordered based on information in the message: the process identifier of the sender, a sequence number, and a logical timestamp at the time the message is sent. The process identifier is guaranteed to be unique in this group. The sequence numbers preserve the order of the messages sent by the application. The logical timestamp is based on a Lamport clock [11], and is used to preserve the causality between messages in the group. These three values are used by a deterministic algorithm applied at the individual receivers to produce a group-wide ordering of messages. Determining message order during a membership change is the work of the protocols that guarantee virtual synchrony (see Section 6).

The reliability service in the InterGroup protocols is represented by the reliable multicast module. The ordering protocols receive messages in sequence number order for each individual source from the reliability service. In the case that an unreliable delivery service is requested, the reliability service is bypassed.

These mechanisms allow the delivery service provided to the application to be chosen, independently, at each receiver. Our approach requires all of the active senders in the group to subscribe to the reliable group timestamp ordered delivery service[4] to preserve causality between messages. This requirement results in unnecessary overhead if none of the processes receiving messages in the process group has subscribed to the reliable group timestamp ordered delivery service. However, the benefits are the flexibility and data abstraction that this method provides.

A benefit of the receiver-oriented selection of delivery service is that the number of participants in the acknowledgment and cut gathering operations can be reduced. Processes that have not subscribed to the reliable group timestamp ordered delivery service do not, in general, have to participate in these operations.

6 Membership

The process group membership is used to update the group membership and allow the delivery of messages to continue after faults, joins and merges.

The InterGroup group communication system takes a novel approach to providing consistent message ordering and delivery within groups. A cornerstone of this approach is the recognition that the message order and reliability constraints can be met by counting only the processes currently sending messages in the group. In the InterGroup system, not all processes are equal. In each process group, a process is classified by its recent activity. If the process has been sending data to the group recently, it is classified as an *active sender*. Each group thus has two memberships; the *receiver membership* that contains all the members of the group, and the *sender membership* that contains only the active senders. The sender group membership is maintained using consistency-based membership mechanisms and is explicitly known. The receiver group membership does not need to be maintained explicitly for the purposes of message ordering and reliable delivery.

The active senders run a membership repair algorithm (MRA) that is based on the membership algorithms used by Transis [2]. The MRA of the InterGroup protocols has been designed so that participation of processes not in the sender group is minimized. The active senders run the membership algorithms to reach a

[4] This does not affect the application requested delivery service. It is handled internal to the InterGroup protocols.

consistent decision on the new membership, to ensure that a unique cut is chosen, and to decide on the place in the message flow that a membership change occurs.

Of the remaining processes, only the processes that have requested the reliable group timestamp ordered delivery service participate in the membership repair. They run a membership repair algorithm built specifically for the InterGroup system, the receiver membership repair algorithm (RMRA).

The RMRA is started when a process receives a message that signals the beginning of the MRA at a process that is an active sender. The process running the RMRA halts delivery of messages to the application and sends out the timestamp of the last message delivered to the application before it stopped delivering messages. This timestamp provides the earliest logical time at which the process can begin a new membership. It then waits for a message that describes the membership change. This message is sent by an active sender and signals the completion of the MRA. The information in this message includes a list of active senders in the new membership and the logical time at which the membership begins. This process attempts to recover and order all of the messages that precede the new membership. If it succeeds in the recovery, it installs the new membership, and successfully completes the RMRA. Otherwise, it does not successfully complete the RMRA.

The combination of the MRA and RMRA allows the active senders to provide virtual synchrony information to the entire group, while keeping the number of participating processes to a minimum.

InterGroup also provides voluntary mechanisms for processes to enter and leave the sender group. A process, wishing to join the sender group, contacts a member of the sender group that serves as a *sponsor*. The sponsor sends a message to the group requesting to add this process to the sender group. The delivery of that message signifies the addition of the process to the sender group. A process wishing to leave the sender group sends a message to the group, requesting that it be removed from the sender group. The delivery of that message signifies the removal of the process from the sender group.

Detailed membership algorithms and their interface to the other modules can be found in [4].

7 Conclusion and Future Work

The goal in designing the InterGroup protocols has been to provide the application services of group communication systems in a wide-area environment with a large number of participants, prone to large latencies and frequent faults, such as the Internet. Thus, we designed an architecture that divides the system into four major components: a control information service, a reliability service, a delivery service, and a membership service. Prior research into scaling the first two of these components allowed us to focus our efforts on the delivery and membership services.

The membership protocols of existing group communication systems have traditionally limited their scalability. The InterGroup protocols take a novel approach towards enhancing the scalability of membership protocols, while maintaining consistent message ordering and delivery within groups. In most applications, only a few group members are sending messages at any one time. A cornerstone of the InterGroup approach is the recognition that the message order and reliability constraints of a group communication system can be met by keeping only the processes currently sending messages in the group membership. The membership protocols require only the sending processes to participate in expensive group-wide decisions.

We step away from the traditional approach of choosing a delivery service to provide more flexibility to the application by allowing each process to choose a delivery service independent of the other processes in the group. This approach also allows processes that cannot meet the desired system quality of service to participate in the group, using a weaker delivery service, and improves scalability of the system.

We are currently undertaking simulation studies of the various aspects of the protocols in order to determine the scalability bounds. We are also measuring the performance of our implementation in order to investigate the performance characteristics. We are expecting to release an implementation of the InterGroup protocols soon, for further testing and use by applications.

References

[1] D. A. Agarwal, L. E. Moser, P. M. Melliar-Smith, and R. K. Budhia. The Totem multiple-ring ordering and topology maintenance protocol. *ACM Transactions on Computer Systems*, 16(2):93–132, May 1998.

[2] Y. Amir, D. Dolev, S. Kramer, and D. Malki. Transis: A communication subsystem for high availability. In *Proceedings of the 22nd IEEE International Symposium on Fault-Tolerant Computing*, pages 76–84, New York, NY, July 1992.

[3] Y. Amir, L. E. Moser, P. M. Melliar-Smith, D. A. Agarwal, and P. Ciarfella. The Totem single-ring ordering and membership protocol. *ACM Transactions on Computer Systems*, 13(4):311–342, November 1995.

[4] K. Berket. *The InterGroup Protocols: Scalable Group Communication for the Internet*. PhD thesis, Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA, 2000.

[5] K. P. Birman and R. Van Renesse, editors. *Reliable Distributed Computing with the Isis Toolkit*. IEEE Computer Society Press, 1994.

[6] V. G. Cerf and R. E. Kahn. A protocol for packet network intercommunication. *IEEE Transactions on Communications*, 22(5):647–648, May 1974.

[7] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6):784–803, December 1997.

[8] I. Foster and C. Kesselman, editors. *The Grid, Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, Inc., 1998.

[9] K. Guo and L. Rodrigues. Dynamic light-weight groups. In *Proceedings of the 17th IEEE International Conference on Distributed Computing Systems*, pages 33–42, Baltimore, Maryland, May 1997.

[10] I. Keidar, J. Sussman, K. Marzullo, and D. Dolev. A client-server oriented algorithm for virtually synchronous group membership in WANs. In *Proceedings of the 20th IEEE International Conference on Distributed Computing Systems*, pages 356–65, Taipei, Taiwan, April 2000.

[11] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, July 1978.

[12] L. E. Moser, Y. Amir, P. M. Melliar-Smith, and D. A. Agarwal. Extended virtual synchrony. In *Proceedings of the 14th IEEE International Conference on Distributed Computing Systems*, pages 56–65, Poznan, Poland, June 1994.

[13] L. E. Moser, P. M. Melliar-Smith, R. K. Budhia D. A. Agarwal, and C. A. Lingley-Papadopoulos. Totem: A fault-tolerant multicast group communication system. *Communications of the ACM*, 39(4):54–63, April 1996.

[14] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications. IETF Request for Comments: 1889, January 1996.

[15] P. Sharma, D. Estrin, S. Floyd, and L. Zhang. Scalable session messages in SRM using self-configuration. Technical Report 98-670, USC, February 1998.

[16] R. van Renesse, K. Birman, M. Hayden, A. Vaysburd, and D. Karr. Building adaptive systems using ensemble. *Software: Practice and Experience*, 28(9):963–979, July 1998.

[17] R. van Renesse, K. P. Birman, and S. Maffeis. Horus: A flexible group communication system. *Communications of the ACM*, 39(4):76–83, April 1996.

[18] B. Whetten, T. Montgomery, and S. Kaplan. A high performance totally ordered protocol. In *Proceedings of the International Workshop on Theory and Practice in Distributed Systems*, pages 33–57, Dagstuhl Castle, Germany, September 1994. Springer-Verlag.