A Network Architecture for Reliable Process Group Communication

Seiji Murata
seiji@mt.cs.keio.ac.jp

Atsushi Shionozaki
shio@mt.cs.keio.ac.jp

Mario Tokoro*
mario@mt.cs.keio.ac.jp

Department of Computer Science, Keio University
3-14-1, Hiyoshi, Kohoku-ku, Yokohama, 223, JAPAN

Abstract

Many applications in distributed systems consist of a set of processes over multiple network nodes. Traditionally, the communication to a set of processes or a process group has been implemented by sending unicast messages to each member of the group, causing unnecessary network traffic. In this paper, we present the design and implementation of the Process Group Management Protocol (PGMP) which provides efficient and reliable process group communication using IP multicast. We also present problems faced with the design of IP multicast through our experience with PGMP, and propose a network architecture that provides efficient and reliable process group communication best suited for distributed applications.

1 Introduction

In distributed systems, many applications, such as multimedia teleconferencing and replicated databases, consist of a set of processes over multiple network nodes. Such a set of processes, which becomes the abstraction for the unit of processing, is called a process group. Each member of a process group cooperates with other members and possibly receives requests from processes that do not belong to the group. Thus, communication involving members of a process group occurs very frequently. Traditionally, messages were sent to each member of a process group using the unicast feature of the underlying network. In this method, application programmers had to construct their own communication routines which manage group membership, thus programs became complicated. Also, programs were inefficient, because identical unicast messages were repeatedly sent. Consequently, a communication system that separates group management from each application and sends messages to all member processes efficiently, which is also easy to use, is needed.

*Also with Sony Computer Science Laboratory Inc. 3-14-13 Higashi-Gotanda, Shinagawa-ku, Tokyo, 141, Japan

In this paper, we present the design and implementation of the Process Group Management Protocol (PGMP) based on IP multicast. PGMP is a transport protocol that provides reliable and efficient process group communication on top of the Internet Protocol. Furthermore, we also present problems faced with IP multicast through our experience with PGMP and propose a network architecture that supports efficient and reliable process group communication for building distributed applications.

The next section discusses existing process group communication systems. Section 3 describes the design of PGMP and in Section 4, its implementation is presented. In Section 5, the performance measurements of our prototype implementation of PGMP is given. In Section 6, a process group communication network architecture which is best suited for distributed applications is presented. Finally, we state future work and conclusions in Section 7.

2 Existing Process Group Communication Systems

In this section, we present existing process group communication systems, and discuss their problems.

ISIS [4, 5] provides process group communication features in a framework for building fault-tolerant distributed systems. ISIS tries to guarantee the correctness of group behavior when failures occur, and maintain efficient delivery of messages. It provides high level group communication functionalities, such as delivering messages while assuring causal ordering or total ordering among messages sent to the same group, not suited for implementation at lower layers of the network architecture. In practice, ISIS implements its functionalities by sending unicast messages to each process belonging to the group, thus a large overhead is incurred on the underlying networks.

Other protocols, such as MTP (Multicast Transport Protocol) [2] and VMTP (Versatile Message Transaction Protocol) [6], also provide process group communication features. These systems aim to pro-

vide efficient process group communication, so they use the multicast mechanism provided by the underlying network. However, these systems simply use the multicast mechanism and add few functionalities, so message delivery to all processes belonging to the group is not guaranteed. These protocols do not take failures into account, nor do they consider the range of nodes the processes in the group encompass. Also dynamic allocation of process group addresses to new created groups is not supported. Therefore, these protocols are not suitable for distributed applications such as replicated databases and file systems, which require reliable message delivery.

To provide an efficient process group communication facility, it should use network multicast, and be implemented as a transport protocol directly on top of it for efficiency. Furthermore, functionalities for creating groups and changing membership dynamically, and send messages to all processes belonging to the destination group reliably, is required. If an application requires additional functionalities such as maintaining the order among messages from different hosts to the same destination group, as ISIS provides, these functionalities can be supported by providing libraries on top of this transport protocol.

3 The Design of PGMP

The Process Group Management Protocol (PGMP) is a multicast transport protocol based on IP multicast, which supports reliable message delivery to all processes belonging to a destination group using message streams[1]. First, we briefly outline the features of IP multicast and the assumptions made of the underlying network, and then describe the design of PGMP.

3.1 Underlying environment

3.1.1 IP Multicast

IP multicast[7] is an extension of IP to support message delivery to all member hosts that belong to the same host group. It provides the following features:

- Host group management
  Allows host group membership to change dynamically and senders to send messages without knowledge of individual members.
- Best-effort message delivery with DVMRP
  Does not guarantee that messages are always received by the destination host.
- Limited host address space
  Uses Class D IP addresses to represent host groups, which are statically allocated from a global IP address space.

Since IP multicast does not provide reliable message delivery, it must be supported along with group management facilities in a transport protocol. Next, we will describe how process groups are managed in PGMP, and how to resolve other problems such as the limited number of class D addresses and bounding the range of message forwarding.

3.1.2 Assumption of the underlying network

A practical transport protocol must be effective under the presence of failures. In an actual internetwork, hosts may crash, messages may be lost, duplicated, or arrive out of order, and link failures may disable hosts from communicating with each other. However, to maintain group processing, each member must be able to communicate with each other individually, and if it is not possible to communicate to a certain host, this host must be considered to be a failed state by all members. If a dynamic routing scheme is used, this assumption is not unreasonable.

3.2 Process Groups

3.2.1 Group Name and Group Address

In PGMP, each process group has a group name and a group address which have a one-to-one correspondence relation. A group name is an identifier used by users, and a group address is used for processing within PGMP. PGMP handles the mapping of a group name to a group address, therefore a user need not know the group address allocated to a process group.

The group names and group addresses must be unique in a range where PGMP is used. But, it is very difficult to guarantee the uniqueness of an identifier for each group in the scope of the entire world. So in PGMP, the range in which group members exist, which we call the *domain* of the group, is divided into a *topological hierarchy*, as shown is Figure 1, and a group name consists of a *service name* specified by a user and a *domain name* representing the range of a group. Thus, in this method, a user only needs to name a process group uniquely within the same domain. Since it is inefficient to incorporate the group name within each packet header for comparison at the receiver, the group name is internally mapped to a group address which also consists of two parts, a domain part statically allocated to each domain, and an identifier part allocated dynamically when a group is created.

For example, the group name for a process group of a teleconferencing application which is used within the Department of Computer Science at Keio University in Japan, shown in Figure 1, consists of `teleconference` (*service name*) and `cs.keio.japan` (*domain name*).

[1] In a stream, the structure of each message is preserved, and sequential message delivery to all processes belonging to a destination group is guaranteed.

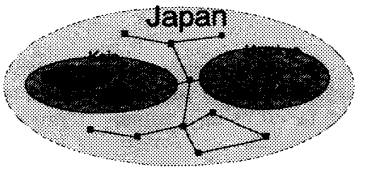

Figure 1: An example of domains in process groups

3.2.2 Dynamic Multicast Address Allocation

Since PGMP uses IP multicast, each process group must be allocated an IP multicast address. For efficient message delivery, each IP multicast address allocated to a process group should be unique within the range of message delivery. If not, a host which is not a member of the destination group may receive a message. In traditional IP multicast, an address is statically allocated to each application, but this method is not suitable for handling process groups which exist for only a short duration, because the number of IP multicast addresses is limited.

We introduce an *address allocator* for each domain to allocate IP multicast addresses dynamically[2]. It allows us to create as many process groups as the number of IP multicast addresses given to its domain without allocating all of them in the traditional IP fashion, appropriate for process groups which exist for only a short duration.

3.2.3 Bounding Range of Message Forwarding

In IP multicast, a message to a host group is forwarded along the multicast tree rooted from the source host. This multicast tree is constructed by cutting branches from the broadcast tree when the first message is broadcast to the network. Usually, since the range in which group members exist is not known, a message must be broadcast to determine the range over which group members exist. In this case, redundant costs for forwarding messages to unnecessary networks and maintaining the information in the branches where no group member exists arise.

As mentioned previously, in PGMP the domain of a process group represents the range in which group members exist. Each process that wants to create a process group, decides the range and selects its domain. Thus, it is possible to decide the scope of a message to prevent forwarding to unnecessary networks from a group name or a group address. Furthermore, it is possible to allocate the same IP multicast address among domains whose range do not overlap. This solves the problem of the limited number of IP multicast addresses that can be used.

However, it is difficult to decide the range of a process group for each domain and forbid messages forwarding across boundaries. In PGMP, the range of process groups for each domain and the length of the longest path between process group members, which we call the *diameter* of the group, is statically set on each domain. PGMP then bounds the range of forwarding messages by specifying a *time-to-live* field, which is as long as the diameter of the process group, in the IP header. In our protocol, messages may be forwarded across boundaries, but the scope of message forwarding is bounded statically. Another method to bound the forwarding area is to improve the routing policy of IP multicast to represent forwarding areas as those provided in HDVMRP[13]. In this case, a domain corresponds to an IP multicast forwarding area, and messages are not forwarded across boundaries, so messages are delivered more efficiently. But, an implementation of HDVMRP was not available to us to exploit this routing policy at this time.

3.3 Membership Management

In order to determine whether group members exist on other hosts when a member leaves a group[3], and send messages reliably based on positive acknowledgement with retransmission, group membership information, called the *group-view*, is maintained at each host. In PGMP, IP multicast is used for host to host communication, and additionally a message is delivered to all local processes in the destination group within a host. We assume delivery within a host is reliable, so we only consider reliability of host to host communication over the networks. Therefore, a group-view consists solely of information of remote hosts on which group member processes exist. If a process is going to join (or leave) the group on a host where other members do not exist, it must send a request message to hosts in the group-view for adding (or removing) the host to (or from) the group-view. Therefore managing group-views becomes complicated. In this section, as examples of group management, the steps required to create and join a group are described.

3.3.1 Creating Process Groups

When a process issues a create new group request, the following steps are taken in PGMP.

1. The requesting process sends a request message with a group name (`ADDR_REQUEST`) to the address allocator by using IP multicast[4], to obtain a group address and an IP multicast address. Next, the address allocator managing the requested domain accepts the request. The address allocator allocates an unused *group address* and an unused IP multicast

[2]Maintaining multiple address allocators per domain for fault tolerance is beyond the scope of this paper.

[3]If there are no members on other hosts, this group must be destroyed.

[4]The IP multicast address of the host group that the address allocator belongs to is statically allocated from the global address space.

address given to the domain, and sends a reply message (**ADDR_REPLY**) to the requesting process. If an unused address is not available within the domain, the request will be rejected.

2. When the requesting process receives **ADDR_REPLY** from the address allocator, it creates a group information table that contains the allocated group address, IP multicast address, a group-view, and other group status information.

Since members of a newly created group do not exist yet, the group-view of other hosts need not be processed.

3.3.2 Joining Process Groups

Figure 2 shows the steps required for joining a process group. As mentioned previously, joining or leaving process groups involves the changing of group-views on other hosts. It is very difficult to maintain consistency of group-views among these hosts when join or leave requests are issued from different hosts simultaneously. In PGMP, by guaranteeing the total order of the acceptance of group-view update request messages among the hosts where group member processes exist (we will call these *member hosts*), consistency of a *group-view* is maintained. The following steps are taken for a join primitive.

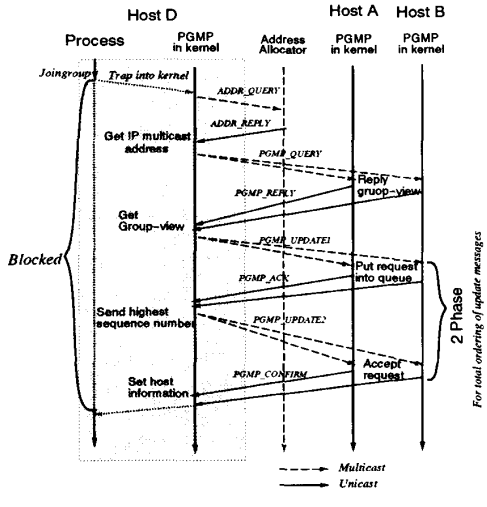

Figure 2: Joining a process group

1. If there are other processes belonging to the same destination group on the host where the process issuing a join request exists, this host has been registered in the group-view on other member hosts. Therefore, **joingroup** simply adds the requesting process to the group information table and returns. If not, the requesting process sends a request message (**ADDR_QUERY**) to the address allocator for querying the address allocated to the target group. The address allocator managing the queried domain searches the requested group from the list of groups within the domain. If found, it sends a reply message (**ADDR_REPLY**) which includes the group address and IP multicast address allocated to the process group back to the requesting process.

   If there is no entry for the queried group, the address allocator returns an error message.

2. The requesting process receives **ADDR_REPLY**, and determines group and IP multicast addresses.

3. Next, the requesting process sends a request message (**PGMP_QUERY**) for querying the current group-view on each member host by using the IP multicast address received in Step 2.

   When a member host receives **PGMP_QUERY**, it replies with its current group-view. The reply message (**PGMP_REPLY**) also consists of a version number of the current group-view. This version number is incremented when the group-view is updated, and used for detecting update request messages based on old group-views.

   Since consistency of the current group-view is maintained throughout all host members, it is only necessary to receive one reply. Thus, **PGMP_QUERY** and **PGMP_REPLY** do not need to be transferred reliably.

4. After the requesting process obtains a group-view from **PGMP_REPLY**, it must update the group-views on other member hosts based on the received group-view.

   If multiple hosts initiate a group-view update simultaneously, some may update host information based on an old group-view. In this case, some member hosts which are not in the old group-view may not be updated. To prevent this, we sort update request message in total order on all member hosts, namely all member hosts accept the update request in the same order, and reject any message based on old group-views. Therefore, consistency of a group-view is maintained. There are many methods for allocating total order to messages, but we use the method which is similar to two-phase commit proposed by Birman and Joseph[4] as follows.

   When the first request message (**PGMP_UPDATE1**) arrives at each host, it is enqueued into the update request queue and marked unacceptable. The receiver then sends an acknowledge message (**PGMP_ACK**) with a sequential number to the source host. If the source host receives **PGMP_ACK** from all member hosts, it sends a second request message (**PGMP_UPDATE2**) with the highest sequence number received. When member hosts receive **PGMP_UPDATE2**, they change the sequence number of the first request message in

the queue to the received number, mark it acceptable, and sort the update request queue by sequence number. If the request message at the top of the queue is acceptable, the host confirms the version number of the request. If it is the current version, the request is accepted, and the hosts updates the group-view, and then replies with a confirm message (PGMP_CONFIRM). If group-view of this request is old, all hosts reject this request and return a reject message (PGMP_REJECT).

5. If the requesting host receives PGMP_REJECT, it repeats from Step 3. If the requesting host receives PGMP_CONFIRM from all member hosts, the procedure for joining a group is completed.

We will elaborate on the case when a member host fails during updating in Section 3.6.

3.4 Reliable Message Delivery

PGMP uses positive acknowledgement with retransmission that involves buffering and retransmision of sent messages. If acknowledgements of message receipts are not received from all member hosts, the message is resent, and the message will eventually be received unless member hosts fail. The following steps are taken when a process sends messages.

The source host puts a message into the send buffer, and sends a message with a sequential message number to member hosts by using IP multicast. At the receiver, received messages are delivered to local processes according to sequential number, the receiver sends an acknowledgement message back to the source host with the maximum message number delivered on this host. When the source host receives an acknowledgement message from the member hosts, it calculates the minimum acknowledgement message number from all member hosts. This number indicates the messages received by all member hosts. Thus, the source host can free such messages in the send buffer. If an acknowledgement message has not arrived from more than one host in the group-view for a given timeout interval, the message is considered lost, and the source host sends the message to the member hosts again. Consequently, sent messages remain in the send buffer and are sent repeatedly until they are received by all member hosts or the source host enters failure mode, as described in Section 3.6.

PGMP also uses a cumulative acknowledgement scheme, so a receiver does not always send acknowledgements back to the source. The receiver sends an acknowledgement after receiving several messages. Since acknowledgement messages are sent by sending unicast messages to the source host, the number of acknowledgement messages seriously affects network traffic. Thus, a cumulative acknowledgement is very effective.

Next, we describe a situation where a new member host joins a group while a messages are being sent to the group and how acknowledgements for these messages are handled. It is necessary to guarantee the delivery of messages issued to a new member host after it joins a group. In PGMP, each member host which has accepted the join request adds the requesting host into the group-view, and sets an acknowledgement number from the requesting host to the current message number. In other words, a host which accepts a join request assumes that the requesting host has already received all messages that it has sent beforehand. Next, it sends a join confirm message back to the requesting host with the current message number. The requesting host receives the confirm message and ensures that only messages with a higher number than the current message number in the confirm message is accepted. If any messages are received by the join requesting host before the confirm message arrives, it ignores them. However, since any host sending such an out of order message will wait for an acknowledgement message from the requesting host, eventually the message will be retransmitted. Therefore, it is guaranteed that messages issued before the join request was accepted are not received, and all messages issued after it are always received and delivered.

3.5 Group-view Notification

To construct higher level facilities on top of PGMP, group membership information must be provided. PGMP provides the mechanism which enable higher level entities to look up current membership information and to be notified of changes in group membership. Using this mechanism, application programmers can construct higher level facilities such as message ordering.

3.6 Fault Tolerance

PGMP assumes that the group-view maintained on each member host is always correct. This assumption is true when failures do not occur, because the consistency of group-views is guaranteed during updates, as shown in Section 3.3. If a failure occurs, some hosts may not be able to communicate, and an acknowledgement message for sent messages or group-view update request messages will never return from such hosts, so group operation will block indefinitely. Such blocking should be handled appropriately.

In PGMP, a failure is detected when an acknowledgements have not been returned within a time bound (it must be long enough to distinguish packet loss or high load of the destination host from failures). If a failure is detected, the detecting host enters failure mode. In this mode, the detecting host gathers the list of currently active member hosts based on its host group-view. This is performed by sending PGMP echo request messages to all member hosts. If the group-view obtained is the same as the old group-view, a

failure did not occur, so failure mode ends. If there is a discrepancy, the host sends a group-view update request message to member hosts. If a host enters failure mode when it is updating its group-view, the previously issued update request message may remain in the update queue on member hosts. If this previous message is not acceptable, it will never become acceptable, so it should be cancelled. If it is acceptable, it will be accepted on all member hosts, so it remains in the queue. It is critical when there are previously issued update request messages from other member hosts in the update queue on each host. However, these messages will either be acceptable or the requesting host for the remaining message will enter failure mode and cancel it. Thus, group-view update request messages issued from a host in failure mode will always be accepted by active hosts. Consequently, the group-view on each active member host will be able to represent the current correct group-view, and group communication can be continued to active hosts.

3.7 Other Useful Functionalities

PGMP provides other useful functionalities not mentioned previously, which include using unicast reply messages for multicast request messages, sending messages from a process that does not belong to the process group, and sending multicast messages with synchronization. These functionalities provide a practical environment for constructing distributed programs that require one to many communication.

4 Implementation

We have implemented PGMP in the BSD/386 kernel on PC-AT compatible machines. The interface libraries shown in Table 1 are provided for application programmers.

Table 1: PGMP primitives

Primitive name	description
creatgroup(sd, sockaddr_pgmp)	create new group
joingroup(sd, sockaddr_pgmp)	join group
leavegroup(sd)	leave group
destroygroup(sd)	destroy group
sendtogroup(sd, buf, length, sockaddr_pgmp)	send message to group
recvfromgroup(sd, buf, length, sockaddr_pgmp)	receive message sent to group

The structure `sockaddr_pgmp` consists of a process group name and address. These libraries invoke Unix system calls such as *setsockopt*(), *sendto*(), and *recvfrom*(). PGMP is implemented in the kernel and requests are processed there. Thus, users can use these libraries transparently without knowledge of how PGMP manages process groups and sends messages.

PGMP provides connection oriented communication, thus connection state must be maintained. When a process group is created, PGMP allocates memory for each group called the PGMP control block (PGMPCB). PGMPCB consists of information on the current connection which includes process group name and address, IP multicast address, host group-view, multicast send buffer, unicast send buffer for each destination host, receive buffer for each source host, etc. If all processes belonging to a group on the local host leave this group, PGMPCB is freed.

The address allocator is implemented as a user level server. It maintains the mapping of group name and group address, and allocated IP multicast addresses on the domain it manages. It also manages the diameter of the domain, and informs it to group members. It receives request messages from the PGMP in the kernel. If the request message is directed to the domain that the address allocator manages, it accepts the request and replies with an allocated group-address and IP multicast address. If not, it will forward the request message to another address allocator which manages a higher or lower domain in the domain hierarchy.

5 Evaluation

5.1 Analysis

Table 2 shows the number of messages required in PGMP compared with traditional systems which use unicast messages to send messages to processes in a process group. In PGMP, by sending messages using IP multicast, we can send a message to all members of a process group by using only one packet per network segment irrespective of the number of processes in the destination group. Thus, we can avoid unnecessary network traffic caused by forwarding identical messages repeatedly, and send a message simultaneously to all members of a process group.

Acknowledgement messages are sent by using unicast messages in both systems, so the minimum number of acknowledgement messages required equals the number of member hosts on which group member processes exist. Therefore, the number of acknowledgement messages seriously affects network traffic. PGMP delays the sending of acknowledgement messages by using cumulative acknowledgement, as described in Section 3.4. Thus, network traffic caused by process group communication is greatly reduced.

Lost messages may be retransmitted by either sending unicast messages to each host, or an IP multicast message to the group. The later is efficient in the case that acknowledgement messages from many hosts have not been received. However, even if there is only one message left to be acknowledged, retransmission messages are sent to all member hosts. In such a case, most member hosts receive an unnecessary message. Therefore, it is difficult to determine one optimal scheme, and PGMP does not define a specific policy.

Table 2: The number of sent messages (N process members existing on M member hosts)

	Traditional: using unicast	PGMP
message send	N (M)	1
acknowledgement	N (M)	maximum: M

5.2 Performance

The performance measurements for PGMP primitives from our implementation in the BSD/386 kernel on 486DX2 (66MHz) PC-AT compatible machines are shown in this section.

First, the execution time for **sendtogroup()** measured by varying the number of member hosts and the size of messages are shown in Tables 3 and 4. These parameters significantly influence the performance of the applications made on PGMP. As shown in Table 4. the percentage of the overhead of PGMP increases with message size, but the main overhead is due to cost necessary for calculating the checksum of the message for reliable message delivery. Checksum must be supported, and presents nearly the same overhead regardless of the layer it is provided. Thus, it can be said that essential costs of PGMP is quite low. Table 3 shows that in PGMP, the time required to send messages to all member hosts is constant irrespective of the number of member hosts. Consequently, we can conclude that the overhead incurred by PGMP is small, and so it fulfills our initial purpose of providing efficient and reliable group communication.

Table 3: **sendtogroup()** execution time (512bytes)

number of member hosts	1	2	3	4
time (msec)	0.40	1.19	1.20	1.20

Table 4: **sendtogroup()** execution time (msec)

size (bytes)	64	128	256	512	1024	4096
PGMP	0.70	0.77	0.90	1.19	1.83	4.20
no checksum	0.60	0.69	0.78	0.97	1.20	2.14
using UDP	0.54	0.63	0.73	0.88	1.12	1.73

Table 5 shows the execution times of other primitives. The internal processing required for **joingroup()** and **leavegroup()** is different, when other process group members exist on the same host from when they do not, as mentioned in Section 3.3.2, so we measured both cases. These costs, particularly for updating group-views, are high. However, they are not used as frequently as send message primitives.

6 Discussion

In this section, we discuss the problems faced with IP multicast through our design and implementation of PGMP, and propose a network architecture best suited to provide efficient and reliable process group communication for distributed applications.

Table 5: Execution time of other primitives (msec)

	creatgroup	destroygroup	joingroup	leavegroup
Local	4.47	4.78	0.12	0.08
Remote	—	—	7.60	3.68

First, IP multicast does not provide group management and reliable message delivery. Thus, PGMP needs to provide group membership management, though IP multicast provides logical addressing to host groups which allows users to send messages to host groups without knowledge of host group membership. If host group membership is managed at the network layer, information about host group topology can be used more efficiency, such as in RM[11].

Second, DVMRP cannot bound the range of message forwarding, resulting in messages being forwarded to unnecessary networks. PGMP controls this by specifying a *time-to-live* field in the IP header. But messages are still forwarded across the group boundaries, so mechanisms which map addresses to forwarding areas, such as those defined in HDVMRP are needed.

The above problems arise because IP multicast was not designed taking process groups into consideration. To provide efficient process group communication, we must consider these requirements and design a network architecture which provides necessary functionalities at the appropriate layers, and in which each layer provides necessary information to upper layers.

We propose the following network architecture.

- Network layer

  This layer should support multicasting facilities of the lower layers, mapping of host group addresses to hardware addresses, group membership management, reliable message delivery between hosts within the logical addressing structure of host groups, and bounding of the range of message forwarding according to host group address.

  Logical addressing allows a transport protocol to send messages without knowledge of host group membership. However group membership information must be provided to the transport layer when necessary. The network layer and the transport layer need to cooperate and provide information to each other to support dynamic address allocation. Particularly, the network layer should provide information about the relationship between forwarding range and network address to the transport layer.

- Transport layer

  This layer should support reliable message delivery between processes and mapping of process group addresses to host group addresses, and if necessary, process group membership information notification to the upper layers should be provided.

  Since the network layer supports reliable interhost message delivery, only the reliable delivery of messages received through networks to group member processes on the local host need be considered.

The host group address allocation policy is defined at the network layer, so if a process group address includes the range of the group, it can be directly mapped to a host group address.

- Application layer

  Services provided at this layer are implemented by user level servers or libraries on the top of the layers mentioned above. This layer must support mapping of process group names to process group addresses and high level group communication facilities such as message delivery with causal and total ordering. This mapping should occur dynamically and support process groups which exist for only a short duration.

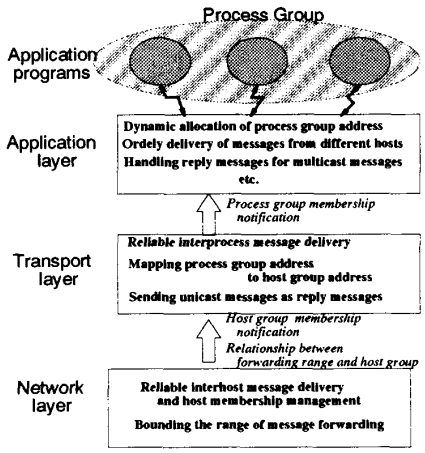

Figure 3: Our proposed network architecture

Functionalities required for process group communication are different for each application, so all of the above is not always necessary. For example, group membership information notification is not required for a teleconferencing application. Therefore, it should be able to select these functionalities dynamically.

7 Conclusions and Future Work

In this paper, we discussed the functionalities necessary in a transport protocol using a network multicast scheme for supporting distributed applications, which are managing process group membership and reliable message delivery. Next, we presented our transport protocol PGMP that provides reliable and sequenced message delivery to all members of the destination process group by using IP multicast. We showed the efficiency of PGMP, and how the overhead incurred by PGMP is small by evaluating our prototype implementation. Thus, PGMP allows users to make programs that communicate with one another efficiently. Finally, we proposed a network architecture that provides efficient and reliable process group communication which is best suited for distributed applications based on our experience with PGMP.

We are presently considering implementing libraries using PGMP to support higher level facilities, such as ordering of messages issued from different hosts [1, 8, 9, 10]. We are also considering methods for rejoining a group separated by network failures. Also PGMP should incorporate other multicast routing schemes [3, 11, 12], such as HDVMRP. Finally, we would like to design a group communication system based on the network architecture described in the previous section.

Acknowledgements

We would like to thank Ken-ichi Murata, Mitsutaka Iwai, and Tatsumi Hosokawa for their comments, and Mitsuaki Suto for his help in setting up the environment necessary for our implementation.

References

[1] R. Aiello, E. Pagani, and G. P. Rossi. Causal Ordering in Reliable Group Communications. In *SIGCOMM '93 Conference Proceedings*, pp. 106–115, Sep. 1993.

[2] S. Armstrong, A. Freier, and K. Marzullo. Multicast Transport Protocol. *RFC 1301*, Feb. 1992.

[3] T. Ballardie, P. Francis, and J. Crowcroft. Core Based Trees (CBT) An Architecture for Scalable Inter-Domain Multicast Routing. In *SIGCOMM '93 Conference Proceedings*, pp. 85–95, Sep. 1993.

[4] K. P. Birman and T. A. Joseph. Reliable Communication in the Presence of Failures. *ACM Transactions on Computer Systems*, Vol. 5, No. 1, pp. 47–76, Feb. 1987.

[5] K. P. Birman, A. Schiper, and P. Stephenson. Lightweight Causal and Atomic Group Multicast. *ACM Transactions on Computer Systems*, Vol. 9, No. 3, pp. 272–314, Aug. 1991.

[6] D. R. Cheriton and W. Zwaenepoel. Distributed Process Groups in the V Kernel. *ACM Transactions on Computer Systems*, Vol. 3, No. 2, pp. 77–107, May 1985.

[7] S. E. Deering. Host Extensions for IP Multicasting. *RFC 1112*, Aug. 1989.

[8] E. Mayer. An Evaluation Framework for Multicast Ordering Protocols. In *SIGCOMM '92 Conference Proceedings*, pp. 177–187, 1992.

[9] H. G. Molina and A. Spauster. Message Ordering in A Multicast Environment. In *9th ICDCS*, pp. 354–361, June 1989.

[10] A. Nakamura and M. Takizawa. Priority-Based Total and Semi-Total Ordering Broadcast Protocols. In *12th ICDCS*, pp. 178–185, June 1992.

[11] B. Rajagopalan. Reliablity and Scaling Issues in Multicast Communication. In *SIGCOMM '92 Conference Proceedings*, pp. 188–198, 1992.

[12] K. Ranvindran and M. Sankhla. Multicast Models and Routing Algorithms for High Speed Multi-Service Networks. In *12th ICDCS*, May 1992.

[13] F. Teraoka and H. Kusumoto. HDVMRP: A Multicast Routing Protocol for Wide Area Networks. In *10th Coference Proceedings Japan Society for Software Science and Technology*, pp. 169–173, June 1993. in Japanese.