Scalable Group Composition with End-to-end Delivery Semantics

Scott Johnson, Farnam Jahanian, and Jigney Shah
Dept. of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI 48109-2122
{scottdj,farnam,jiggey}@eecs.umich.edu

Abstract

This paper examines the problem of building scalable, fault-tolerant distributed systems from collections of communicating process groups while maintaining well-defined end-to-end delivery semantics. Our approach to inter-group communication takes advantage of modular group composition. We introduce a generic model which provides a distinction between intra-group and inter-group communication, and allows for independent modular composition of protocols for each type of communication. Using this model, we provide a framework for analyzing the end-to-end semantic behavior of messages, making it possible to determine exactly what delivery semantics a set of receivers will observe for a given message, even when those receivers are members of different process groups which use different group multicast protocols and enforce different delivery semantics. To realize this model, we present an implementation architecture which is based on a simple abstraction: inter-group routers*. Inter-group routers provide a mechanism for the exchange of messages between process groups. They allow groups to communicate in a fault-tolerant manner without sharing group or message state information and without modifying the underlying communication protocols. Finally, we present simulation results of our architecture for a variety of inter-group topologies and communication patterns. Using the published performance data of an existing group communication protocol, we simulate the performance of our architecture to verify its scalability. We also confirm that the end-to-end semantics predicted by our model are actually enforced by the architecture.*

Keywords: group communication, distributed systems, fault-tolerant multicast, ordering semantics, predictable communication

1 Introduction

Group communication is a widely studied paradigm for building distributed systems. In this approach, a distributed application is structured as groups of cooperating processes built on two key primitives: group membership and fault-tolerant multicast communication. It has been argued that process groups are particularly effective for managing the complexity of large applications and for providing dependability and timeliness guarantees in the presence of faults. A process group can be used, for example, to provide high availability through active replication of system state, to increase performance by partitioning the computational load, or to support rapid dissemination of information from senders to a group of potentially anonymous recipients.

Numerous group communication systems have been developed during the last decade. Totem [1], ISIS [2], Horus [3], Consul [4, 5], Delta-4 [6], Amoeba [7], and Transis [8] are examples of such systems. These systems allow a set of processes to communicate using various delivery semantics such as FIFO, causal, or total ordering while providing a consistent membership view of participants believed to be accessible. A number of group communication systems also provide explicit support for real-time applications.

For example, TTP [9], RTCAST [10] and XPA [11] provide guaranteed response time by exploiting real-time operating system features or special purpose hardware support. Other approaches including the *fail-awareness* framework [12], *quasi-synchronous* model [13], and Cactus [14] are based on the best effort paradigm.

One of the primary motivations for group communication is its use as a building block for providing replication and consistency in distributed applications. As the needs of distributed applications have grown, new protocols and services have been developed to provide greater flexibility and robustness to the application developer. Both Horus and Consul allow micro-protocols to be composed together to provide exactly the functionality required by an application without the overhead associated with unused services. Cactus also utilizes micro-protocols while offering support for real-time distributed applications. Protocols such as Totem, ISIS and RMP [15] each provide several different types of delivery semantic so that messages do not incur unnecessary delay while stronger semantics are enforced. One key challenge that remains to be addressed by group communication services, however, is their scalability as the number of processes in a distributed system increases.

It has been recognized that traditional group communication techniques may not scale well when used in large scale distributed applications. In addition, given the increasing popularity of collaborative systems and other types of distributed applications which involve geographically distributed computers, it is also important to address the performance and scalability of these protocols when used in a regional or wide-area network environment. Existing work in this area has primarily taken two approaches. Some researchers have attempted to improve the scalability of the single process group model [16], and its use in wide-area environments [17, 18] even in the presence of partitions [8]. Others have designed architectures which allow systems to be composed from multiple process groups while maintaining some type of delivery semantic on messages sent between groups [19, 20]. The latter approach allows the size of each process group to be kept small enough to ensure good performance within the group, while still allowing large numbers of processes to be used in a single system. In this paper, we build on these architectures to design a generic framework for composing process groups modularly and scalably, with arbitrary end-to-end delivery semantics and flexible composition topologies.

In our experience, many distributed applications have an inherent structure which maps very well into a system composed of multiple process groups. Processes which cooperate on a particular task or form a specialized component within an application are natural candidates to be placed into a separate process group. For example, if an application executes several distributed computations in parallel, each one could be performed in its own process group, with each group possibly residing on its own local-area network. This would allow standard methods of replication and consistency checking to be enforced within each computation, but without the resource contention and extra overhead that would be incurred if they were all placed in a single process group or on a single LAN. Furthermore, it should also be possible to utilize different group communication protocols for different groups, so that each group's particular needs can be met with minimal extra overhead. For example, one group in a distributed application might employ Totem for its Safe Delivery guarantee, while another might simply require the causal delivery of Horus' causal broadcast service, while yet another might use RTCAST for total ordering with real-time delivery guarantees. Provided the process groups can communicate with each other, and can enforce some type of end-to-end delivery semantic on inter-group messages, this type of system can realize the same functionality as the single-group model, but with potentially better scalability and greater flexibility. These types of techniques might also provide better performance to process groups which are distributed over a wide-area network, since each group protocol can be run using only local processes, while still allowing semantically robust communication with groups in other

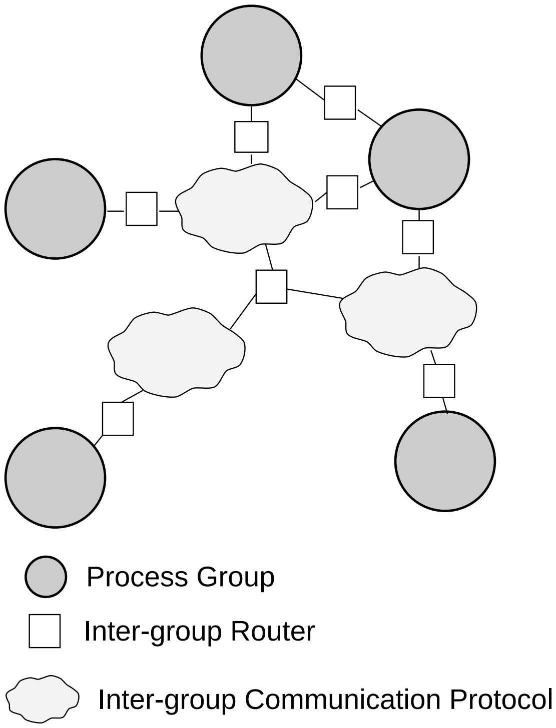


Figure 1: Example of a group composition using inter-group routers

locations. This would allow the strong service guarantees provided by group communication to be practically applied to a much larger class of distributed system problems.

In order to realize the group composition model, we propose a new architecture, the *inter-group router* architecture, which supports modular composition of process groups while maintaining well-defined, flexible end-to-end delivery semantics. The inter-group router serves as both a logical and a physical bridge between two or more communication protocols. The inter-group router operates at a higher layer in the protocol stack than the communication protocols it connects, re-sending messages it receives from one protocol in the other protocols based on the message's set of destination groups. To each connected protocol, it appears that an application is simply sending messages to other local recipients. When used in conjunction with process groups, the inter-group routers allow messages to be forwarded between groups over an inter-group network without requiring the group communication protocols themselves to be aware of the existence of the other groups or to share any state information with them. We will show how inter-group protocols with appropriately chosen delivery semantics can be used to connect the process groups and inter-group routers while enforcing desired end-to-end delivery semantics over all messages, regardless of which process group each message was sent from. Figure 1 shows an example of a group composition using inter-group routers.

Even in a system which satisfies these properties, there are still several important issues which must be addressed. First, unlike the clearly defined delivery semantics of single-group multicast protocols, messages exchanged by composed process groups may not exhibit consistent, easily determined delivery semantics end-to-end, especially when they are semantically related to other messages which are never seen by the destination group. In addition, the failure semantics of composed systems are likely to be more complex than those observed in a single-group system, and the inter-group routers must be fault-tolerant to prevent network partitions and message loss. The remainder of this paper addresses these issues in more detail.

Section 2 presents a generic model for inter-group communication and describes the architecture of the inter-group router. Section 3 gives a detailed analysis of the end-to-end semantic properties of inter-group messages in group compositions. Section 4 examines the failure semantics of group compositions and shows how inter-group routers can be made fault-tolerant to prevent paritions and message loss. Finally, we compare our approach to those of previous multiple group communication schemes in section 5, and show simulation results to validate our claims of scalability and semantic guarantees in section 6.

2 Basic Group Composition Model

We assume that a distributed system is composed of multiple process groups which communicate to perform application-level tasks. Groups may be distributed across multiple LANs and multiple groups may co-exist on each LAN. Each process may belong to more than one group, and each group may send and receive messages from any other group in the system. Each message may be sent to one or more process groups, or to a subset of the processes within each group if the group communication protocols permit it. We call the process group which includes the sender of a message the *local* or *sending* group, and all process groups which contain receivers of a message *remote* or *receiving* groups. This distinction will make it easier to describe the behavior of inter-group messages when we look at their end-to-end delivery semantics.

We divide communication in the group composition model into two types: *intra*-group (or local), and *inter*-group (or remote). In intra-group communication, a *local* or *intra-group* message is sent and delivered within a local group using only the local group's multicast protocol. Messages which include receivers in both local and remote groups are considered to be inter-group messages. In inter-group communication, a message is sent from one process group to another via one or more inter-group communication protocols, which operate over one or more physical inter-group networks. The inter-group protocols may or may not be group multicast protocols; however, both the local and inter-group protocols should provide a well-defined message delivery semantic, such as sender-based FIFO, Causal, or Total ordering. This communication model has the benefit of separating intra- and inter-group communication, which we will show greatly simplifies the enforcement of various message delivery semantics. However, this model requires a mechanism for forwarding messages between process groups and the inter-group network.

This model of communication is very similar to the design of standard computer communication networks, where processors often communicate with other processors that reside on physically separate networks and which may even use different types of communication media and protocols. In these systems, no processor can send a message directly to a processor on a separate physical network, and often the sender may not even know the remote processor's physical location or address. In order to allow multiple physical networks to be connected into larger systems, a separate message forwarding mechanism, called a router, is often used. Routers forward traffic between otherwise separate phsyical networks, even ones using completely different protocols, and provide an abstraction of the remote networks to each sender. Using these devices, processors are able to communicate with remote hosts without having to physically join their network or use the same protocol.

The same abstraction can also be applied to realize communication between process groups and inter-group communication protocols. With respect to inter-group communication, process groups in

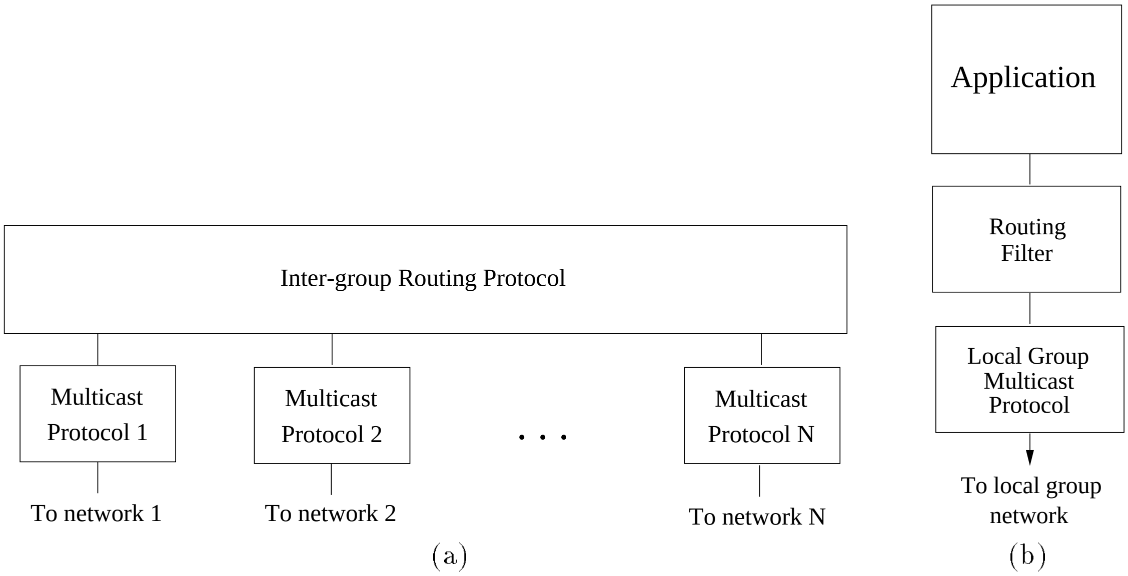


Figure 2: Protocol stack for (a) inter-group routers and (b) group members

our model function much as LANs do in the computer network environment. We therefore introduce the concept of *inter-group routers* as a means of realizing inter-group communication. Inter-group routers provide both a physical and a logical connection between two or more process groups. An example of a system composed using inter-group routers is shown in figure 1.

Inter-group routers do not distinguish between intra-group and inter-group communication protocols; they simply provide a logical connection at the application level between these entities for the purpose of routing and forwarding messages between them. Since they operate at the application level, inter-group routers can leverage existing fault-tolerant group communication services to realize inter-group communication with little additional overhead. Using inter-group routers, each process group or communication protocol may be placed on a separate physical network to improve performance by reducing contention for network resources. For groups and protocols that reside on the same physical network, inter-group routers serve as a logical connection, allowing senders and receivers in each protocol to communicate transparently without requiring the protocols themselves to communicate directly.

The software architecture of the inter-group router is shown in figure 2a. It simply consists of a protocol layer that runs as an application on top of all connected communication protocols. Each application (i.e. non-router) node has a small filter inserted in its protocol stack just below the application layer as shown in figure 2b. This filter exports a multi-group addressing scheme to the application, and is responsible for making sure that messages intended for other groups are delivered to the appropriate local inter-group routers through the local communication protocol. It may also insert header information to help the inter-group router forward messages appropriately.

The inter-group routers are a logical construct. Physically, they can be used to connect networks and forward messages between process groups residing on those networks. For composed groups residing on a single network, the inter-group router is connected to the network and is a member of each group (figure 3a). It forwards messages by receiving them from each group and resending them over

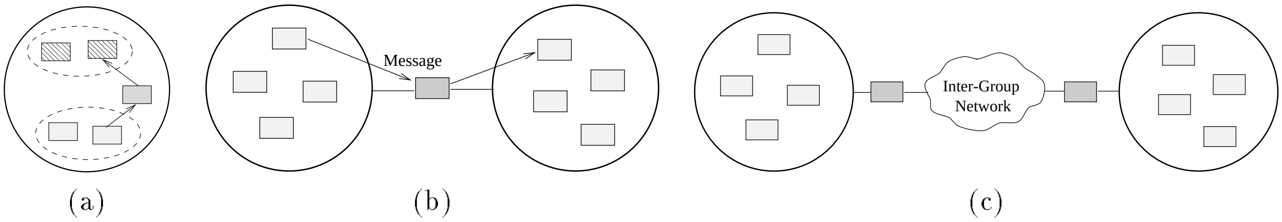


Figure 3: Physical connection of process groups using inter-group routers: (a) groups and router located on same network, (b) two groups on different networks connected directly through a router, (c) two networks connected by an inter-group network

```
send(msg, destination_list)
   if (destination_list includes processes in other groups)
      local_destination_list = local processes in destination_list
            + all inter-group routers attached to the local group
      routing_header = non-local destination address(es)
      prepend routing_header to msg
   else
      local_destination_list = local processes in destination_list
   multicast msg to local_destination_list using local group protcol
end
```

Figure 4: Inter-Group Routing Filter Operation (called by applications)

the network using whatever inter-group delivery semantic is required by the application. Similarly, composed groups which reside on multiple networks in the same geographic location may be connected directly to a single router which forwards messages between the networks and enforces the inter-group delivery semantics (figure 3b). In both of these cases, the inter-group router also serves as the inter-group communication protocol. Finally, each group or network may be connected by an inter-group router to a physically separate inter-group network (figure 3c). An inter-group communication protocol is run over the inter-group network to allow the routers to forward messages to other groups with some semantic guarantee. Process groups running on LANs which are not in the same physical location must use this latter approach, in which case the inter-group network is likely to be some type of wide-area network. Each group and inter-group protocol may be arbitrarily connected by one or more inter-group routers to other groups and inter-group protocols, forming group composition topologies that can be quite complex. Note that each of these compositions is simply a different physical implementation of the same logical model – that of process groups connected by inter-group routers and inter-group protocols.

Message delivery for inter-group messages is performed according to the procedure described in figures 4 and 5. The inter-group routing filter at each application node implements a send function that is used by applications to send messages in the distributed system. The inter-group router simply loops indefinitely, processing each message as it arrives. For messages that are sent by a local application to the local group only, the routing filter at each node simply passes the message through unchanged, and it is delivered normally by the local group protocol. For messages sent to other process groups, the routing

```
while (messages arriving) do
   recv msg from protocol p
   if (msg has already been received from another protocol)
      discard msg
   for (every directly attached group in destination list of msg)
      multicast msg to destination processes in group
   for (every other destination group)
      determine next_hop for destination
      determine outgoing_protocol for next_hop
      add next_hop to next_hop_list(outgoing_protocol)
      add outgoing_protocol to protocol_send_list
   for (each protocol in protocol_send_list)
      multicast msg to next_hop_list(outgoing_protocol)
end
```

Figure 5: Inter-Group Routing Protocol Operation

filter at the sender will insert a header in front of the application message indicating the destination groups' addresses and any other information needed by the inter-group routing protocol. If needed, this header information may include a unique message id which enables the inter-group routers to detect duplicate messages. The message is then sent to any inter-group routers in the local group using the local group protocol. The message is also delivered to receivers in the local group at the same time. The message is delivered to the local receivers and the router exactly like any other application-level message. The router uses the routing header to forward the message via its inter-group protocol stacks to the appropriate inter-group communication protocols based on the message's intended recipients.

Within the inter-group communication network, the inter-group routers forward messages between inter-group communication protocols, making routing decisions about where to forward each message based on its destination and the system toplogy. The information needed to make these routing decisions can be determined using standard routing protocols, with the inter-group communication protocols propagating the routing information between inter-group routers. When the message reaches the destination group, the inter-group router receives it from an inter-group protocol and multicasts the message to the destination processes. To the group multicast protocol the message appears to have originated locally. The routing filter at each receiving node strips off the routing header and delivers the message to the application.

Effectively, the routers are implementing *tunneling* for process group messages which are sent by applications through the inter-group protocols to other process groups. Since the router behaves as a normal sender or receiver in each communication protocol, they do not need to be modified in any way in order to compose them together using the the inter-group routers. This router architecture completely isolates the operation of each local group protocol from events in other groups. This achieves our goal of allowing modular composition of process groups.

An important feature to note about this forwarding mechanism is that messages are forwarded from each communication protocol in the exact order that they are delivered to the inter-group routers by that protocol. This means that whatever semantic ordering is enforced on a stream of messages when they are delivered to a router for forwarding will be preserved when those messages are sent in the outgoing communication protocol. We call this the **FIFO forwarding mechanism**, and it is crucial for making end-to-end semantic guarantees about inter-group message delivery.

It is also interesting to note that functionally, there is no difference between an inter-group communication protocol and a process group protocol. Each one takes messages from a collection of local senders, and delivers those messages to a collection of local receivers in some semantic order. The only important difference between the two is that, from the perspective of the entire system, inter-group communication protocols do not generate new messages, but only forward existing messages between other protocols.

This section presented the main elements of our group composition mechanism: the inter-group router architecture, the separation of communication into intra- and inter-group messages, and the FIFO forwarding mechanism realized by the inter-group routers. We now examine the end-to-end delivery semantics of messages in systems composed from multiple process groups, using the the inter-group FIFO forwarding mechanism to enforce strong delivery guarantees even on messages from multiple process groups.

3 Analysis of End-to-End Delivery Semantics in Group Compositions

As we have previously mentioned, one of the key challenges when composing distributed systems from multiple cooperating process groups is determining end-to-end message delivery semantics, especially when the system is composed of different types of group multicast protocols each of which may provide a different delivery semantic. In this section we present a detailed examination of this problem for several types of composition topologies and show how they may be combined to create more complex systems. Throughout this section, we will use the semantic definitions given in [21]. In our terminology, sender-based FIFO (or just sender-FIFO) order is equivalent to their FIFO broadcast, our causal order is equivalent to their Causal Broadcast, and our total order is equivalent to their FIFO Atomic Broadcast (though we will also consider their Atomic broadcast, where totally ordered messages are not delivered in FIFO order).

We begin our semantic analysis by examining a simple composition consisting of a single pair of process groups. After analyzing the basic principles observed in this composition, we determine what kinds of semantic guarantees can be enforced when additional groups are added to the system. Next, we consider the semantics of pipelines of communicating process groups. Finally, we show how to combine these compositions to form more general topologies, including ones suitable for use in wide-area networks.

3.1 Paired Group Composition

We begin our analysis with a paired group composition: a system which is composed of just two groups joined by a single inter-group protocol. This composition illustrates many of the basic issues affecting inter-group communication semantics, and provides the foundation for a more general analysis. Figure 6 provides a graphical depiction of the paired group composition.

This composition can also be viewed as a *client-server* group composition, and is a commonly used architecture in distributed systems. For example, one of the groups may implement a replicated server, with a pool of clients residing in the other group and accessing the server over the inter-group network. In the graph, nodes represent process groups, and the directed edges represent application-level communication patterns between the groups.

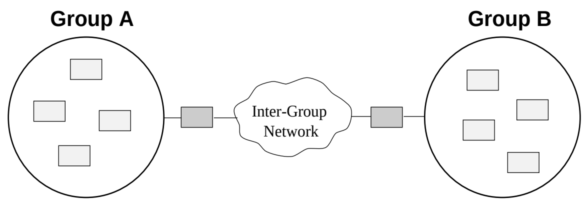


Figure 6: Paired Group Composition

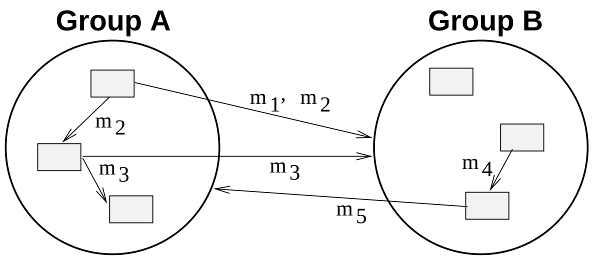


Figure 7: Example of causal delivery in a paired group composition.

Since the intra-group protocols do not need to be modified when composed together using inter-group routers, the delivery semantics of messages sent only within a single process group will remain unchanged. We therefore confine our analysis to the case where messages are being exchanged between the groups. There are two cases to consider: the semantic ordering of messages sent by processes in only one group and delivered in both groups, and the relative semantic ordering of messages sent by processes in both groups and delivered in both groups. The following example illustrates a possible communication scenario.

Example: Consider a paired group composition in which both groups enforce causal delivery on messages (see figure 7). The inter-group protocol is assumed to be sender-based FIFO. Suppose a process in group A sends messages m_1 and m_2 to group B. In addition, m_2 is also sent to a process in group A, which sends causally dependent message m_3 to processes in both groups. A process in group B receives m_3 and sends a causally dependent message m_4 to another process in group B. That process then sends a causally dependent message m_5 to group A. Although for clarity's sake the figure does not show it, messages sent by each process to the other group are considered to be delivered to all processes in that group.

Since causal delivery is inherently sender-based FIFO, m_1 will be delivered before m_2 in both groups thanks to the FIFO forwarding mechanism. Within group A, m_3 will be delivered after m_2 by A's causal delivery protocol. Within group B, m_3 will be sent by the inter-group router after m_2, and so will be delivered after causally preceding m_2 at all receivers. Group B's causal delivery protocol will ensure that m_4 is delivered after m_3 at all receivers within group B. Finally, m_5 is sent to group A. Within group A, m_5 will appear to be sent by the inter-group router after it has received causally preceding message m_3, and so m_5 will be delivered after m_3, even though group A never saw m_4. Thus, all messages in this example are delivered in causal order at all receivers in both groups, regardless of the location of the original sender of each message. $\diamond$

We can see from the example that the inter-group routers provide an abstraction of each process group, making it appear as a single sender or receiver to other groups in the system. Furthermore, notice that because of the FIFO forwarding mechanism, the delivery semantic of each group is preserved on that group's messages when they are delivered to receivers in other groups. We foralize this propety with the following lemma:

Lemma 1 (Paired Single Sender): Consider a paired group composition where the inter-group protocol enforces sender-based FIFO delivery semantics, and both process groups enforce at least sender-based FIFO delivery. The set of messages sent by processes in the *same* process group will be delivered with that group's delivery semantic at all receivers in both process groups.

Proof: The proof of this follows from the definition of sender-based FIFO and our inter-group routers. By definition, messages are always delivered in the sender's group with the delivery semantic provided by that group's multicast protocol. The messages will be received by the inter-group router at the same time, which will forward them to the inter-group network protocol in their original delivery order. Finally, the sender-based FIFO delivery semantic of the other protocols will maintain that order until the messages are delivered to the receiving processes in the other group. □

This is a fairly intuitive result; we are simply taking a predetermined sequence of messages and forwarding them in FIFO order to the receivers. The situation becomes more complicated when we consider the relative semantic ordering between messages which originate in different process groups. The following lemmas show what type of end-to-end delivery semantics can be guaranteed depending on the delivery semantic that is provided by the process group protocols.

Lemma 2 (Paired FIFO Delivery): Given a paired group composition, assume processes in both groups send messages destined for receivers in both groups. If each group enforces sender-based FIFO delivery, and the inter-group protocol also enforces sender-based FIFO delivery, then the delivery semantic observed for all messages at all receivers will be sender-based FIFO.

Proof: This follows directly from the definition of FIFO delivery. Each protocol delivers messages in sender-FIFO order, and that ordering is preserved by the FIFO forwarding and delivery of each succeeding protocol. □

Once again this property is pretty intuitive. With sender-based FIFO, messages from each sending process are not ordered with respect to messages from other processes. Since each protocol enforces sender-based FIFO, and messages are always forwarded between protocols in FIFO order, there is no chance for them to be reordered incorrectly.

We next consider a more complicated case, where messages are delivered end-to-end in causal order.

Lemma 3 (Paired Causal Delivery): Given a paired group composition, assume processes in both groups send messages destined for receivers in both groups. If each group enforces causal delivery, and the inter-group protocol enforces sender-based FIFO delivery, then the delivery semantic observed for all messages at all receivers will be causal.

Proof: Causal delivery can be defined in terms of Lamport's happens-before relation [22], which we denote by $\rightarrow$. Table 3.1 lists the different ways two messages can be causally related, and the delivery order that must be observed if causal delivery is to be enforced. Details of the proof appear in the appendix.

Let "$\text{send}_p(m_i)$" denote the event where process p sends message m_i, and "$\text{recv}_p(m_i)$" denote the event where process p receives message m_i.

For all messages m_i and m_j and processes p and q, m_i must be delivered before message m_j if [23]:

1. $\text{send}_p(m_i) \rightarrow \text{send}_p(m_j)$ (i.e. p sends m_i then sends m_j)
2. $\text{recv}_p(m_i) \rightarrow \text{send}_p(m_j)$ (i.e. p receives m_i and then sends m_j)
3. If $\text{recv}_p(m_i) \rightarrow \text{send}_p(m_k)$ and $\text{recv}_q(m_k) \rightarrow \text{send}_q(m_j)$ (i.e. p receives m_i and sends m_k, then q receives m_k and sends m_j)

Table 1: Properties of causal dependence

This lemma shows that we can achieve a global causal ordering on messages in the system with only a sender-based FIFO inter-group protocol. Furthermore, because of the inter-group routers the groups do not need to share any protocol or message state information to achieve this result, making the composition much more scalable than putting all of the processes in a single group.

Unfortunately, we can't get total ordering in quite the same way. Consider the following scenario: A process in group A sends inter-group message m_1 at the same time that a process in group B sends message m_2. Message m_1 will be delivered in group A and forwarded by the inter-group router to group B. Message m_2 will be delivered in group B and forwarded to group A. Depending on the relative timing of the message send operations, group A could see the delivery order $m_1 \rightarrow m_2$ while group B sees the delivery order $m_2 \rightarrow m_1$, without violating any of the protocols delivery semantics. Obviously, as long as messages are delivered first in the sender's group and second in the receiver's group, this scenario could occur. To get around this problem, we can modify the message forwarding scheme slightly to require that messages which must be totally ordered with respect to both groups must be delivered in the inter-group protocol *first*, in a total order to both inter-group routers. Only then can it be forwarded to and delivered in the process groups. We will refer to this method of forwarding messages as *totally ordered message forwarding*. Naturally, this scheme requires that the inter-group protocol enforces totally ordered message delivery. Although this may seem to be a significant overhead for realizing a global total ordering on messages, it is really no slower than the previous forwarding scheme. The sending group can no longer immediately deliver totally ordered messages, but the end-to-end latency is unchanged.

The total ordering semantics are summarized in the following lemma:

Lemma 4 (Paired Total Delivery): Given a paired group composition in which the inter-group routers perform totally ordered message forwarding, assume processes in both groups send messages destined for receivers in both groups. If each group enforces total ordering, then all messages will be delivered at all receivers in total order.

Proof: Proof of this lemma follows directly from the totally ordered message forwarding scheme and the FIFO forwarding mechanism of the inter-group routers. Each message is forwarded to the inter-group protocol, where it is delivered to the inter-group routers in total order. The message is then

Semantics of:			Effective
Sending Prot'l	Inter-Group Prot'l	Recv Prot'l	End-to-End
S	Any >= Sender-based FIFO	S	S
Total	Total w/ first delivery	Total	Total

Table 2: End-to-End Delivery Semantics in Paired Group Compositions

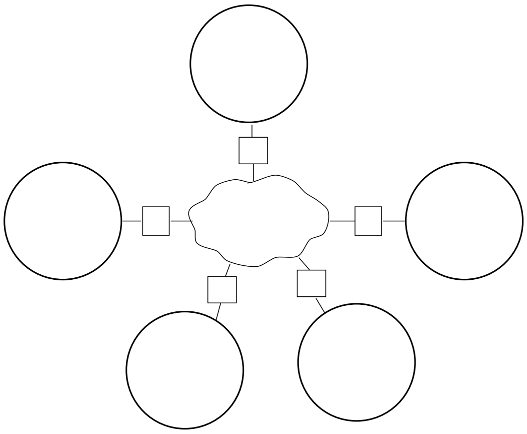


Figure 8: Multi-group Composition

forwarded to the process groups in the same total order. Since the protocols support sender-based FIFO delivery, all messages will be delivered in the same total order at all receivers in both groups. □

A summary of the semantic properties of the paired group composition is given in table 2.

3.2 Multi-Group Composition

Although the paired group composition can be useful for certain types of applications, it is possible to compose much more complex systems while enforcing strong end-to-end delivery semantics. We now extend the paired group composition model by composing multiple process groups using a single inter-group protocol. A typical example is shown in figure 8.

Unlike the paired group case, a simple, sender-based FIFO inter-group protocol does not provide very strong inter-group ordering semantics for this type of composition. It is still possible, however, to obtain robust, flexible end-to-end delivery semantics when the inter-group protocol enforces stronger ordering semantics.

For purposes of our discussion, assume that the group composition consists of a set of groups $G = \{g_1, g_2, \ldots, g_n\}$ connected by an inter-group protocol I. We consider a set of messages M which are required to be semantically ordered with respect to each other. There may be other, semantically unrelated messages in the system which are not delivered in a semantic order with respect to the set M. The process groups which contain senders of inter-group messages in M belong to the set of sending groups, $G_S \subseteq G$, and those groups which contain receivers of inter-group messages in M belong to the set of receiving groups, $G_R \subseteq G$. Each group g_i enforces some well-known delivery semantic S_{g_i} on messages sent within g_i, and the inter-group protocol I enforces well-known delivery semantic S_I on messages sent within I.

We now extend the results of the paired-group semantic analysis to the multi-group composition. We consider three types of communication in the multi-group composition: one-to-many, many-to-many, and many-to-one.

3.2.1 One-to-many Communication

First, we consider a set of messages M that originate within a single process group and are sent to an arbitrary set of receivers. Since messages are delivered to the inter-group routers using the sending group's delivery semantic, and are then forwarded in FIFO order to all receivers, the messages in M will be delivered with the sending group's delivery semantic at all receivers. We formalize this property in the following lemma, which is an extension of lemma 1 to include multiple receivers.

Lemma 5 (One-to-many): Given a multi-group composition, assume that the set of messages M is sent by members of a single process group g_s, to an arbitrary set of receivers. Assume also that the inter-group protocol enforces sender-based FIFO delivery semantics, and that every process group in the system other than g_s enforces sender-based FIFO delivery. If g_s enforces delivery semantic S_{g_s}, then all messages in the set M will be delivered to all receivers with the semantic S_{g_s}.

Proof: The proof of this lemma is available in the appendix.

This is a simple but very useful result. Note that since we are not enforcing a semantic order between messages from different process groups, the set of messages from each sending group can be considered independently. Based on this property, if a distributed application does not need to enforce a semantic ordering between messages from different groups, then it can be composed using a sender-FIFO only inter-group protocol, such as TCP, resulting in a very simple and efficient system design. Also, note that almost every stronger delivery semantic includes sender-based FIFO. It is included in the definition of causal delivery, and although in some definitions of total order it is not enforced, most existing implementations of totally ordered delivery do enforce sender-based FIFO as part of their total order.

3.2.2 Many-to-many Communication

We now consider the case where some semantic ordering is required to be enforced between messages from different process groups. First, we look at the simple case where the application(s) require only sender-FIFO delivery. This is really just a special case of lemma 5, since sender-FIFO by definition only orders messages from each sender independently. Formally,

Lemma 6 (Multiple sender FIFO Delivery): Given a multi-group protocol composition, assume that each process group enforces sender-based FIFO delivery, and the inter-group protocol also enforces sender-based FIFO delivery. In this case, all messages will be delivered in sender-based FIFO order at all receivers.

Proof: Proof of this lemma follows directly from lemma 5, with D_{g_s} = sender-based FIFO. □

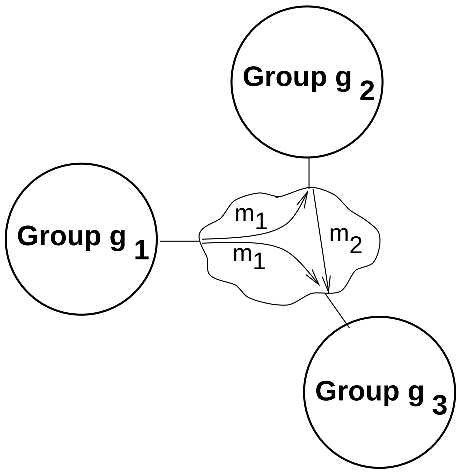


Figure 9: Example of bad causal delivery with sender-based FIFO inter-group protocol (for simplicity, the inter-group routers are not shown)

For a stronger delivery semantic, we now consider the case where causal ordering must be enforced on messages from two or more process groups. Consider the example shown in figure 9. Group g_1 sends message m_1 to groups g_2 and g_3. Group g_2 then sends causally dependent message m_2 to group g_3. Message m_1 and m_2 can be delivered in either order at group g_3 without violating the delivery semantics of a sender-based FIFO inter-group protocol. However, if the inter-group protocol enforces a causal ordering on message delivery, it will not deliver m_2 until m_1 has been delivered. We now prove that this will hold for any number of senders and receivers assuming an inter-group protocol that enforces causal ordering. This is similar to the basic topology of the causal daisy architecture [20].

Lemma 7 (Multiple Sender Causal Delivery): Consider a set of messages M which are sent within a multi-group composition by a set of sending groups G_S to a set of receiving groups G_R. G_S and G_R may overlap. Every group in G_S enforces causal delivery, and every group in G_R enforces at least sender-based FIFO delivery. If the inter-group protocol enforces causal delivery, then all messages in M will be delivered to all receivers in an order that respects the causal precedence relationships within M.

Proof: The proof of his lemma is avaiable in the appendix.

The next case we consider is when a total delivery order must be enforced at all receivers for a set of messages originating at different senders. There are two subcases of this delivery semantic to consider. First, consider a set of messages where at most one sending group also receives inter-group messages (i.e. $|G_S \cap G_R| \leq 1$). In this case, the problem we observed with total delivery in the paired group composition can not occur, since no two groups send messages to each other. We can therefore achieve a global total order on messages using the standard inter-group router forwarding mechanism and an inter-group protocol that supports total delivery. We prove this property in the following lemma:

Lemma 8 (Multiple Sender Total Delivery): Consider a set of messages M which are sent within a multi-group composition by a set of sending groups G_S to a set of receiving groups G_R, such that

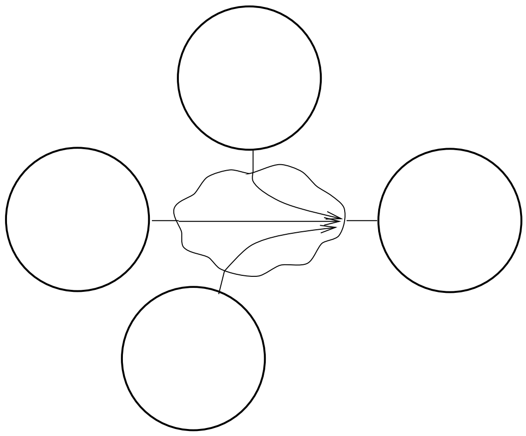


Figure 10: Many-to-one communication in a multi-group composition

$G_S \cap G_R \leq 1$. Each group in G_S must enforce total delivery, and each group in G_R must enforce sender-based FIFO delivery. Messages are forwarded normally (i.e. the special totally ordered forwarding scheme is not used). If these properties hold, and the inter-group protocol enforces total delivery, then all messages will be delivered in total order at all receivers.

Proof: The proof of this lemma is available in the appendix.

If, two or more process groups send totally ordered messages to each other ($|G_S \cup G_R| > 1$), then we face the same problem that the paired group composition had with total delivery. Fortunately, the same solution we used in the paired-group composition can be used to prevent this problem. We formalize this property with the following lemma:

Lemma 9 (Alternate Multiple Sender Total Delivery): Consider a set of messages M which are sent within a multi-group composition by a set of sending groups G_S to a set of receiving groups G_R. G_S and G_R may overlap. Every group in G_R must enforce at least sender-based FIFO delivery of messages, and the inter-group protocol must enforce total delivery. All messages are delivered in the inter-group protocol before being delivered to receivers in the sending group. If these properties hold, and the inter-group protocol enforces a total order on messages, then all messages from all senders will be delivered in a total order at all receivers.

Proof: The proof of this lemma is available in the appendix.

Once again, achieving a global total ordering on messages within the system may seem to have a very high cost, but remember that the end-to-end latency across groups doesn't change; it simply takes longer to deliver them in the sender's group than it does with normal message delivery.

3.2.3 Many-to-one Communication

In many-to-one communication, many process groups send messages which are destined only for processes in a single, common receiving group (see figure 10). In this case, message delivery is much simpler since many of the possible semantic violations simply can't happen. Total ordering, for example,

Semantics of:			Effective
Sending Prot'l	Inter-Group Prot'l	Recv Prot'l	End-to-End
Sender-based FIFO	Sender-based FIFO	Sender-based FIFO	Sender-based FIFO
Causal	Sender-based FIFO	Sender-based FIFO	Sender-based FIFO
Causal	Causal	Sender-based FIFO	Causal
Causal	Total	Sender-based FIFO	Sender-based FIFO
Total	Sender-based FIFO	Sender-based FIFO	Sender-based FIFO
Total	Causal	Sender-based FIFO	Sender-based FIFO
Total	Total	Sender-based FIFO	Total

Table 3: Basic End-to-End Delivery Semantics in a Multi-group Composition

Semantics of:			Effective	
Sending Prot'l	Inter-Group Prot'l	Recv Prot'l	End-to-End	Condition
S. FIFO	Total	S. FIFO	Total	$G_S \cap G_R = \emptyset$
S	S. FIFO	S	S	One sending group (one-to-many)
S	S. FIFO	S	S	One receiving group (many-to-one)
Total	Total w/ early delivery	S. FIFO	Total	$\|G_S \cap G_R\| > 1$

Table 4: Special Cases of End-to-End Delivery Semantics

can be easily achieved if the inter-group and receiving group protocols are only sender-FIFO, since there is a single receiving router in the inter-group protocol.

Lemma 10 (Many-to-One Delivery): Consider a set of messages M which are sent within a multi-group composition by a set of sending protocols G_S to a single receiving protocol g_r. Each message in M may also be delivered to processes in the sending group. Assume every sending protocol enforces some delivery semantic S. If $g_r \in G_S$ then g_r must also enforce S, otherwise it may support just sender-based FIFO. If the inter-group protocol supports just sender-based FIFO in such a system, then the set of messages M will be delivered with delivery semantic S at all receivers.

Proof: The proof of this lemma is available in the appendix.

A summary of the semantic properties of the multi-group composition is given in table 3. Special cases of inter-group communication are summarized in table 4. The lemmas we have proven in this section allow us to build group compositions using a multi-group communication protocol while providing strong, end-to-end semantic guarantees. However, relying on a single inter-group protocol for our group compositions may hinder scalability, since the overall scalabiliy of the system would be dependent on the scalability of the inter-group protocol. We therefore need a way to combine multi-group compositions together scalably to build still larger systems. The remainder of this section discusses how groups can be composed in this way to form pipelines and general compositions of process groups.

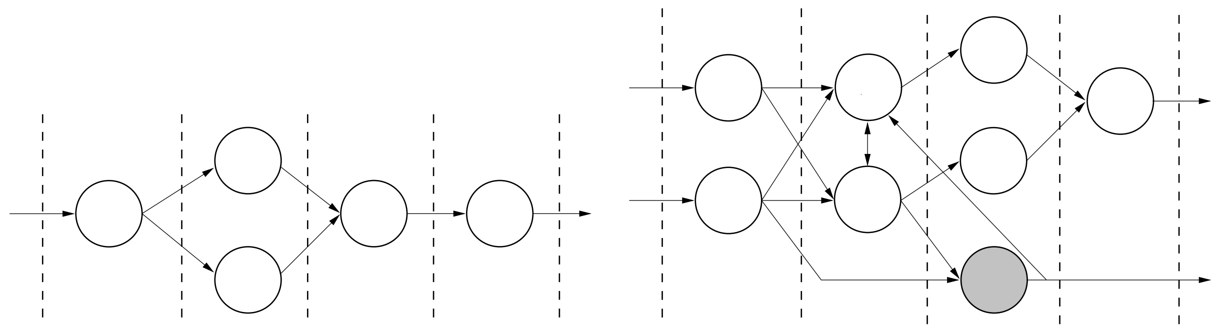


Figure 11: Examples of pipeline communication patterns

3.3 Pipeline Group Composition

The pipelined group composition is a slightly more complex composition structure than the multi-group composition. Instead of connecting groups by a single inter-group protocol, there can be an arbitrary number of groups composed together using multiple inter-group communication protocols. However, pipeline compositions still exhibit a degree of structure in the way groups communicate that allows us to make strong, end-to-end semantic guarantees about message delivery.

We informally define a pipeline to be any group composition where there is a well defined set of input groups, a well defined set of output groups, and communication generally takes place in a direction from the input groups to the output groups. A pipeline stage consists of a set of groups that receive messages exclusively from a set of groups P which form the immediately preceding pipeline stage, and send messages exclusively to a set of groups S that form the immediately succeeding pipeline stage. Furthermore, the preceding and succeding stages should not overlap; that is, $P \cap S = \emptyset$. Groups within the same pipeline stage may also communicate freely with each other. This is similar to the notion of Vertex Separators in [18]. Note that this definition of pipelines is intentionally vague; it simply defines a general class of group compositions. We will provide more formal definitions for some specific types of pipelines and their semantic properties below.

It has been recognized that many distributed real-time applications can be constructed as pipelines. Examples include command and control, automated manufacturing, and air traffic control systems. Many applications accept some data as input and perform a series of computations on that data to produce a final result. Each stage of the pipeline might be used for a different phase of the computation, where the group communication services of each group are used to replicate data for both performance and reliability. Results are then passed from one phase to the next through the inter-group communication protocols, using minimal communication with other pipeline stages.

Some examples of pipelines are shown in figure 11. As with previous figures, the directed arrows show the communication paths between different groups. The vertical dashed lines in this figure show the stage divisions in each pipeline, though in some cases there may be more than one way to define the stage devisions. For example, the shaded group in Figure 11(b) may be considered a part of either the second, third, or fourth pipeline stage. As the examples show, one of the benefits of the pipeline structure is that when group communication takes place in stages it greatly reduces the number of message sources for each receiver. Note that with this structure, it is possible to use a separate inter-group protocol between each pair of pipeline stages, since messages sent between each pair of stages can be delivered independently of messages in other stages. This has the benefit of reducing the size of each inter-group protocol, making the system's scalability dependent more on the size of the pipeline stages than the actual number of groups or processes in the system.

For purposes of our semantic analysis, we do not consider the case where the process groups themselves forward messages between two external groups. Although this might seem like a potentially serious omission, consider that any composition which requires messages to be forwarded between process groups can be re-composed as an inter-group protocol with the process group directly attached.

We have identified three specific classes of pipeline that provide additional ordering constraints on message traffic. Applications will be able to use much simpler inter-group protocols, by exploiting the message ordering automatically created by these pipeline structures. The following sections give a formal description of each class of pipeline and its effects on inter-group message traffic.

3.3.1 Implicitly Causal Pipelines

The first class of pipelines we examine is structured in such a way that inter-group messages which are causally related can never be received out of causal order, assuming a FIFO delivery channel between each group. Each process group must still enforce a causal ordering on messages sent within the group, however. Formally, we can prove that this holds for any pipeine which satisfies the following definition:

Definition 1: An *implicitly causal pipeline* is any composition of process groups such that for every pair of groups A and B, if group A communicates directly with group B, there is no other directed communication path from A to B.

Lemma 11 (Implicitly causal pipeline): If a composition of process groups satisfies the definition of an *implicitly causal pipeline*, all inter-group messages sent within the pipeline will be delivered in causal order at all recipients, provided that all groups within the pipeline enforce causal ordering on messages sent within the group, and the inter-group protocol enforces a sender-based FIFO order on messages sent between groups.

Proof: The proof of this lemma is available in the appendix.

The structure of this type of pipeline is very similar to the causal separators described in [18]; certain process groups act as causal separators that partition the composition into independent stages and prevent the propagation of causal events to other parts of the system.

Obviously, not every application that requires causal ordering can be composed as an implicitly causal pipeline with purely sender-based FIFO inter-group delivery. However, once the communication pattern of the groups has been determined, it is possible to use sender-based FIFO inter-group delivery for sections of the pipeline that do conform to the implicitly causal pipeline definition, and causal inter-group delivery for those sections that don't. Using this technique, it is possible to transform any pipeline into one that guarantees end-to-end causal ordering. Due to space constraints we do not present the details of this algorithm.

3.3.2 Implicitly Totally Ordered Pipelines

In addition to the implicitly causal pipeline, there is another class of pipeline which provides similar benefits when inter-group messages are required to be totally ordered. An implicitly totally ordered pipeline can be defined in a similar way.

Definition 2: An *implicitly totally ordered pipeline* is any composition of process groups such that no pair of sending groups communicates directly with more than one of the same receiving groups.

Lemma 12 (Implicitly totally ordered pipeline): If a group composition satisfies the definition of an implicitly totally ordered pipeline, all messages sent between groups are delivered in total order to all receivers, provided all groups in the system guarantee total ordering of local messages and the inter-group protocol guarantees sender-FIFO ordering.

Proof: The proof of this lemma is available in the appendix.

As with causal delivery, it is possible to convert any pipeline into one that guarantees end-to-end total ordering. The only change is the condition by which a set of groups must be composed with the same inter-group protocol.

3.3.3 Implicitly Causal-Total Pipelines

Finally, some applications may require that inter-group messages be both causally and totally ordered. By combining the implicitly causal and implicitly totally ordered pipeline definitions, an implicltly causal-total pipeline can be defined.

For this type of group composition, the follow property is guaranteed:

Lemma 13 (Causal-total pipeline): If a composition of process groups satisfies the causal/total pipeline definition, then all messages sent between groups will be delivered in causal and total order at all recipients, provided that the local group protocol guarantees causal and sender-FIFO total ordering, and the inter-group protocol guarantees sender-FIFO ordering.

Proof: This follows directly from the proofs for implicitly causal and totally ordered pipelines. □

3.4 General Group Composition

Although the paired group, multi-group, and pipeline group compositions can be used to design a wide variety of distributed systems with multiple, cooperating process groups, they still require that group communication in the application exhibit certain patterns of communication. In a general group composition, we consider a set of groups that communicate in arbitrary patterns using multiple inter-group protocols which are connected to form an inter-group network of arbitrary topology.

As before, we consider a set of messages M which are semantically related to each other, but which are unrelated to other messages in the system. Messages in M are forwarded through the inter-group network by the inter-group routers over some subset of the possible paths in the actual physical topology; we call this the *routing toplogy* for M. We can greatly simplify the semantic analysis for general

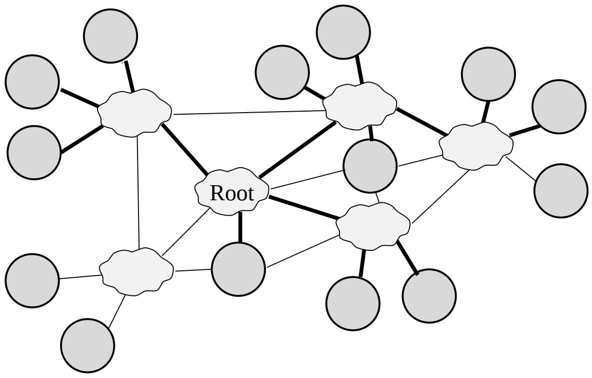


Figure 12: Sample tree-based routing topology for a given set of messages M in a group composition (inter-group routers are not shown). Links which are part of the routing topology are shown in bold. Note that not all groups are connected to the tree; only groups that send or receive messages in M are included.

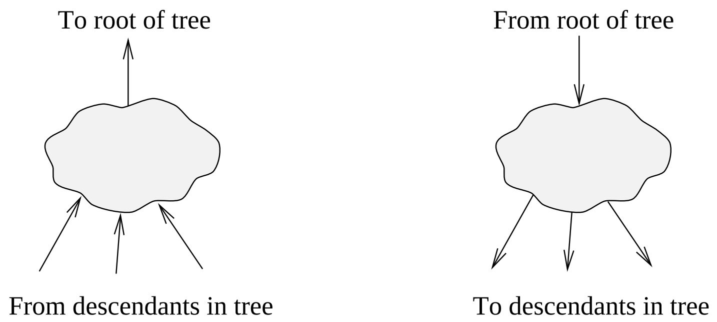
compositions by considering only the routing topology instead of all communication paths in the system. For example, Figure 12 shows a sample routing topology for one set of messages where many physical links of the system are unused. Since messages sent over paths which are not in the routing topology are by definition not in M, the remaining physical topology will not affect the delivery of messages and can be safely ignored.

As an example of how separating the physical topology from the routing topology can simplify message delivery semantics, consider a set of messages M which are routed over an inter-group network via a single path between two process groups. If message delivery within the inter-group network is sender-based FIFO, the messages will exhibit the same semantic behavior as messages in a paired group composition, since the routing topology consists of two groups connected by a sender-FIFO inter-group communication protocol.

Within a general group composition, we require that the routing topology for each set of messages M form a tree connecting all of the groups that send or receive messages in M. This routing topology ensures that there will only be a single route between each pair of groups, and that there are no loops in the routing graph. The example in figure 12 uses a tree for its routing topology. Since there is only a single path between each pair of process groups, messages can travel in both directions over each link in the routing topology. The routing topology can therefore be represented by an undirected graph consisting of a subset of the links and communication protocols in the physical topology. Although inter-group routers are not shown in the graph, we assume that inter-group routers are used to connect the protocols as previously described.

This requirement greatly simplifies the enforcement of message ordering semantics. If messages could follow different paths through the inter-group network, any semantic orderings which had already been enforced on them by a protocol would be lost, and could not be reconstructed at the receivers without additional state information. Furthermore, if there were loops in the routing topology, similar problems could occur between semantically related messages that followed different paths around the loop to get to the same receiver.

By repeated application of lemma 5, we can show that all messages from a single sending protocol will be delivered to all receivers with the delivery semantic provided by the sending protocol, as long as



(a) Messages are first routed toward the root of the tree...

(b) and then routed from the root to all destinations

Figure 13: Communication pattern of each inter-group protocol in a core-based tree

the intervening inter-group protocols enforce sender-based FIFO. As with previous compositions, then, the only way messages can be delivered out of order is when messages from two different sending groups must be semantically ordered when delivered to the same receiver(s).

We now describe two different types of inter-group routing trees that can be used to enforce delivery semantics in a general composition. These trees are modeled after the core based tree topology [24] which has been used for IP multicast routing over the Internet.

In the first variation of the core based tree, each message is first delivered in the inter-group protocol at the root (or core) of the tree. It is then forwarded from the root through the tree to all receivers.

In an inter-group routing topology, the core based tree has the advantage of allowing all semantic ordering to be enforced at the root of the tree; all other inter-group protocols need to enforce only sender-based FIFO to ensure semantically correct delivery. Therefore, this topology is particularly well-suited to wide area networks, since each process group and the root protocol of the tree can be run on separate LANs distributed over a wide geographic area, and needing only a sender-based FIFO wide-area network to connect them. Of course, this type of routing topology will not be as scalable, since a single protocol at the root of the tree has to semantically order all inter-group messages. However, this limitation is not as critical as it might appear, since each set of messages M within the application can use a different routing tree with a different root, and it is likely that the number of messages in each set is much less than the total number of messages sent by the application. If this is not the case, then one of the other group composition topologies would probably be more appropriate for composing the application.

It can be shown that message delivery semantics can be enforced using a core based tree routing topology with only sender-based FIFO inter-group protocols. From figure 13, we can see that all communication in the inter-group protocols is either many-to-one or one-to-many. Therefore, by lemmas 10 and 5 they need only enforce sender-based FIFO ordering to preserve the original semantic ordering of messages sent though the tree.

The second varation of the core based tree approach is to simply follow the normal core based tree delivery model: whenever one group sends a message, that group is treated as the root of the tree for purposes of forwarding that message. This results in much more efficient message propagation, since

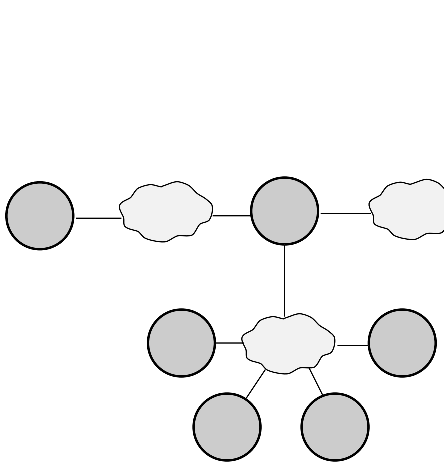


Figure 14: Example of a hybrid composition: a simple, three-group pipeline connects a multi-group composition and a paired composition to form a more complex system

messages do not need to traverse the same link more than once, as can happen with the first approach. This method also does not depend on the centralized ordering used by the first approach, which should improve its scalability. On the other hand, since message ordering is not being done by a single, central protocol, the inter-group protocols will have to support stronger delivery semantics than just sender-based FIFO in order to ensure correct end-to-end message delivery. For each inter-group protocol in the inter-group routing tree, the delivery semantic which must be enforced is determined by the number and type of directly attached protocols which will send and messages in M, and on the end-to-end delivery semantic required by the application. The exact delivery semantic required can be determined from the multi-group composition lemmas.

A final alternative to creating general group compositions is to combine several different composition topologies into a single system. For example, a subset of the groups in an application may map very well into an implicitly causal pipeline, while another subset of groups requires a multi-group composition, and yet another subset communicate as a paired group composition. After composing subsets of groups using these well-defined composition topologies, they can be connected together to form a single hybrid composition (figure 14). This technique enables us to build systems that are much more complex than any one of the pre-defined topologies.

4 Fault-tolerant Inter-group Communication

In previous sections we presented the basic architecture of the inter-group router, which enabled processes to send messages to receivers in other process groups. We also analyzed the effective end-to-end delivery semantics of inter-group messages forwarded by the inter-group routers. We now examine how the inter-group router architecture affects the fault-tolerance of the composed systems. There are two important facets of fault-tolerance which are affected by the inter-group routers. First, we consider the failure semantics of the composed groups. Second, we discuss how the inter-group routers themselves can be made fault-tolerant to prevent network partitions and message loss.

4.1 Failure Semantics in Group Compositions

We have already described how the inter-group router architecture allows communication protocols to be composed modularly to form larger systems. Group protocols are not modified when composed with the inter-group routers, so whatever failure semantics are offered by each protocol will be unchanged with the introduction of the inter-group routers to the group. Failure of the inter-group routers themselves is addressed below.

Since none of the protocols share any state, failures and other types of membership events in one protocol are not propagated to other protocols in the system. Therefore, the operation of each group in the system is unaffected by the failures of members of other groups. Application processes, however, can still be affected by failures in other parts of the system. There are two ways that this can happen. First, the failure of a process or processor in some group can cause an application-level service to become unavailable to other parts of the system. For example, the failure of a processor in one process group may cause part of a replicated database to become unavailable to the rest of the distributed system. However, as with current practice, important components of each group in the composition are likely to be replicated to prevent loss of application-level services from failures such as this.

The second type of failure is one that causes a network partition, thus preventing otherwise correct groups from communicating with each other. For performance reasons, it is very likely that each process group will be contained within a single LAN. Network partitions of a LAN are extremely unlikely, as failures that would partition a LAN would most likely prevent *all* communication within the LAN. To prevent this, most fault-tolerant communication protocols make use of redundant hardware and retransmissions.

The wide-area networks, on the contrary, are much more likley to experience network partitions, especially when inter-group communication takes place over a wide-area network. This is a very difficult problem, and is currently an active area of research. One promising approach involves the use of an enriched view synchrony for processes communicating over asynchronous, partitionable networks [17]. Other solutions which are applicable to fault-tolerant distributed systems may be used to improve the fault-tolerance of group compositions as well. Finally, since the failure of an inter-group router can partition the inter-group network, it is important to ensure that the inter-group routers themselves are adequately fault-tolerant. The following subsection describes how this can be achieved.

4.2 Fault-tolerance of Inter-Group Routers

To improve reliability, many group multicast protocols are used with redundant, fault-tolerant physical networks to reduce the chance of message loss due to network failure. Some examples include the broadcast networks described in [25, 26]. This section describes how inter-group routers might be replicated to achieve a similar level of fault-tolerance for inter-group communication, while maintaining the guaranteed end-to-end delivery semantics.

Recall that an inter-group router is a logical entity. Normally we have assumed that each router was implemented by a single host. We can improve the fault-tolerance of inter-group routers by adding additional physical nodes to each logical router and using replication techniques to reduce the chance that a failure will cause the router to stop forwarding messages. We accomplish this by replicating the incoming and outgoing message streams so that failures will not cause messages to be lost or misordered. This can be done using traditional replication techniques such as primary-backup or active replication. These replication schemes ensure that the routers do not become a single point of failure in the inter-group network, reducing the likelihood that an inter-group router can fail and partition the inter-group network. By using active replication, this can be accomplished without significantly compromising performance. We have developed an optimized active replication scheme for the inter-group routers, but a presentation of it is beyond the scope of this paper.

5 Comparison to Previous Work on Group Composition

In the introduction, we referred to a number of group communication systems that have been developed during the last decade. We also discussed several approaches for improving the performance and scalability of these protocols. This section focuses on several architectures which have been proposed to solve the group composition problem. In this section we compare them to our approach and discuss the relative merits of each.

The Totem Multiple Ring protocol [19] is an architecture based on the Totem single ring protocol [1]. It allows multiple Totem rings to be interconnected by gateways, and enforces a global total order on all messages sent within the system. However, there are several key differences between the Totem approach and the inter-group router architecture.

The Totem multiple ring protocol assumes a different model of how applications use group composition. The Totem system uses a group model in which any process on any ring can be a member of any group in the system. This allows much greater flexibility in the allocation of processes and processors to different groups in the distributed system, but it does incur a much higher cost to maintain the state for each group when its members are not located in the same part of the network. In our approach, the potential membership set of each group is restricted to a single LAN, although multiple groups can co-exist on the same LAN. This restriction is what allows the inter-group router architecture to achieve its goals of modular group composition and decoupling intra-group communication from inter-group communication. Although it is not as flexible as the totem multiple ring approach, we believe that by not allowing groups to be distributed over multiple networks we can achieve a higher degree of scalability while still allowing process groups to communicate freely.

In the Totem multiple ring protocol, the system can only be composed of Totem single rings. With the inter-group router architecture, on the other hand, different types of process groups and other communication protocols can be composed together modularly, based on the communication requirements of the distributed application. Furthermore, the Totem multiple ring protocol always enforces a global total order on all messages for all groups in the system based on a the timestamp ordering of a Lamport clock. This requires the forwarding of each message to be delayed until succeeding messages with higher timestamps from other rings are received. Although this is a very strong semantic guarantee, we believe that more efficient systems can be built if the end-to-end semantic of the system can be chosen based on the application's requirements. With the inter-group router architecture, global total ordering can be enforced when needed, but the end-to-end delivery semantics of the system can also be tailored to meet the application's specific communication needs, allowing more lightweight and efficient protocols to be used when possible. In addition, in wide-area network environments, the inter-group router architecture allows the application designer to compose the system using inter-group protocols that are designed to operate efficiently over wide-area networks.

Another key difference between the two approaches is that in the Totem multiple ring protocol, the gateways that bridge the individual rings can only connect two such rings at a time. Inter-group routers can connect multiple protocols at a time, which reduces the number of networks and communication protocols needed for some topologies and allows for shorter communication paths between communicating groups. Furthermore, in the Totem multiple ring protocol, every gateway and group member must know the entire topology of the system in order to maintain the global total ordering, and any topology changes must be propagated to every node. The inter-group routers are much more like traditional routers in this respect; they only need to know the next hop for a given destination, and group members do not need to know anything about the topology of the system at all.

The Hierarchical Daisy Architecture [20] is another approach to group composition. In some respects it is very similar to the inter-group router approach – each group may be composed modularly and may reside on a separate physical network, and some events in each group do not need to be propagated to other groups. Processes are assumed to belong to only one group. However, the daisy architecture supports only causal inter-group message delivery. It also requires that message delivery and stability information be propagated across groups before messages can be discarded. This type of state propagation may adversely affect scalability and allow failures in one group to adversely affect the performance of other groups. The inter-group router architecture does not require state propagation to achieve the same delivery semantics; messages can always be discarded by a sender after they have been delivered to the inter-group router. Finally, Daisy Architecture requires a strict, hierarchical group composition topology, while the inter-group router architecture can be used to create arbitrary composition topologies.

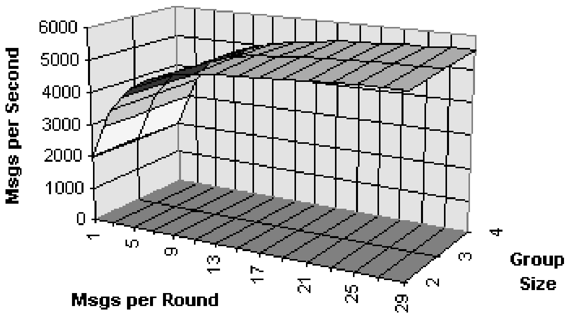
6 Performance Evaluation

In order to evaluate the scalability of the inter-group router architecture, we performed a trace-driven simulation of systems of process groups composed by inter-group routers for varying group sizes and message transmission rates. Our results show that the inter-group router architecture does indeed give some significant performance benefits for large systems, improving both throughput and latency compared to systems of comparable size composed using a single process group.

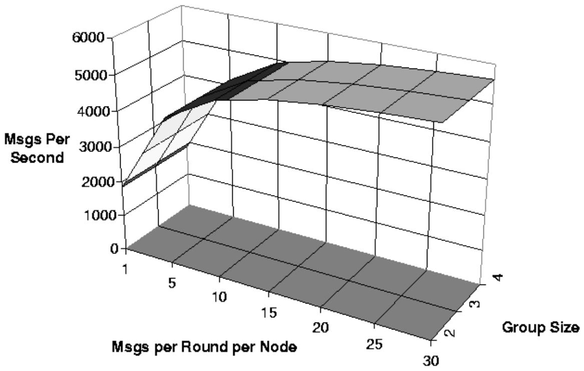
Our simulation was carried out using Opnet, version 3.5.A. To implement a group communication service within the simulation, we used the actual production code for RTCAST [10], a group multicast protocol we have developed which provides predictable, atomic message delivery with real-time guarantees. RTCAST uses a logical token ring to enforce both causal and total ordering on messages delivered to the group, and it totally orders group membership changes with respect to messages. Based on the published performance evaluation of our actual RTCAST implementation [27], we parameterized the protocol stack and tuned the simulation to accurately reflect the real performance of the protocol. Since the performance of RTCAST compares very favorably with the published performance data of other group multicast protocols (most notably Totem [1] and Horus [3]), we expect that our simulation results will be applicable to systems built using other protocols besides RTCAST. Figure 15 shows how the performance of a real RTCAST group implementation compares to an equivalent simulated RTCAST group. The simulation gives performance results matching almost exactly to the actual implementation, indicating that it is an accurate predictor of the behavior in real systems.

To compare the scalability of a single-group vs. a multiple group system, we ran simulations for a single process group, a paired group composition, and a four-way multi-group composition. These topologies are shown in figure 16. To compose the groups, we implemented the inter-group router protocol from figure 5 within Opnet. For the inter-group protocols, we implemented a simple sender-based FIFO protocol for the paired group composition and used RTCAST for the multi-group composition. The simulation environment allowed us to test group sizes much larger than our actual experimental testbed.

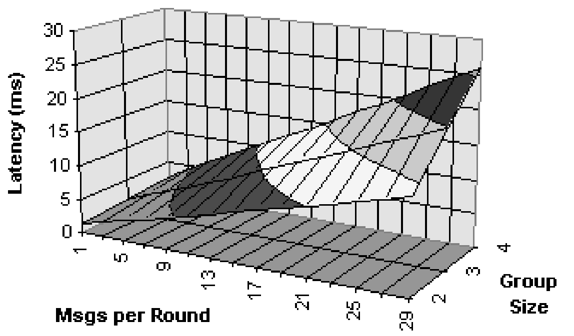
There are two areas of protocol performance that we focused on in our simulations: message latency,



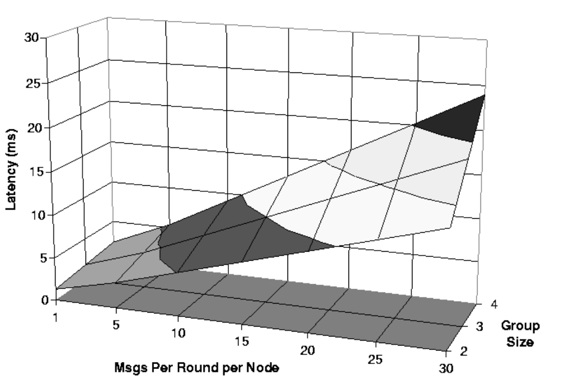
(a) implementation maximum message throughput



(b) simulation maximum message throughput

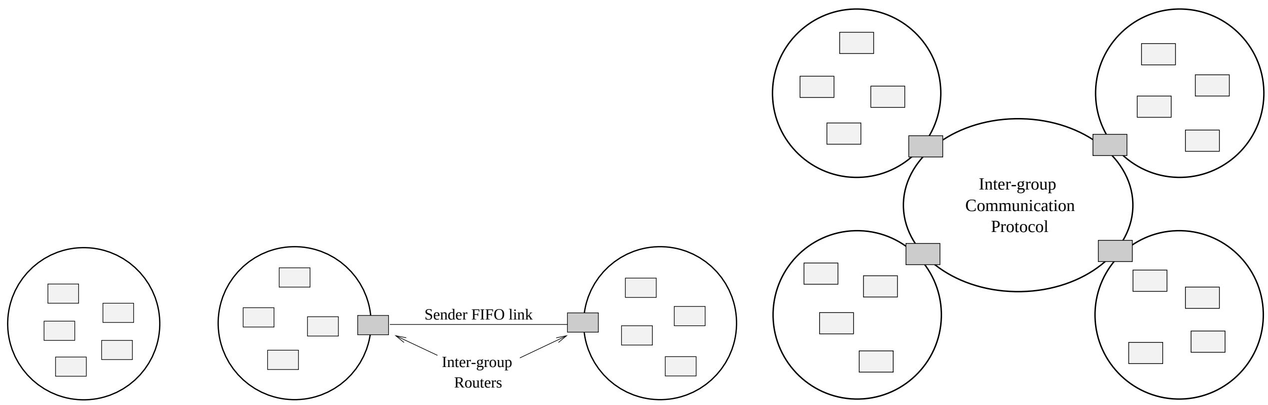


(c) implementation avg. message latency



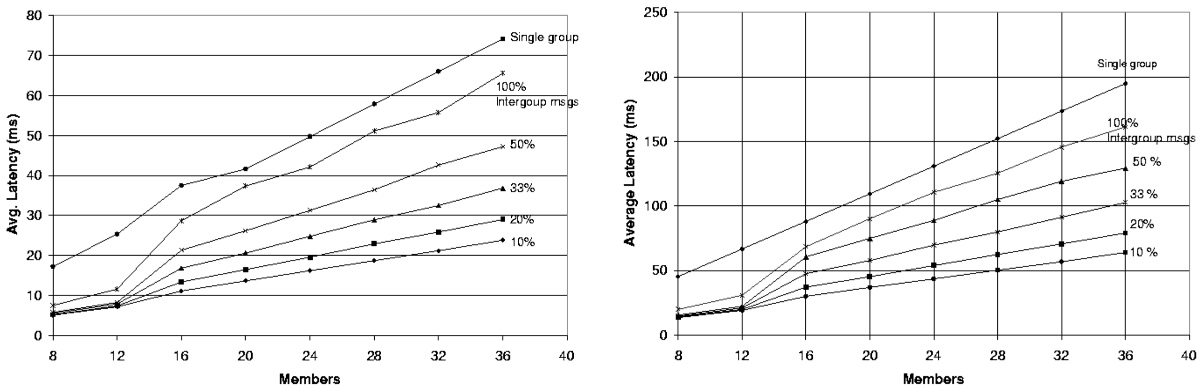
(d) simulation avg. message latency

Figure 15: Comparison of performance for RTCAST implementation and simulation (all tests use 1-byte messages



(a) single group (b) paired group (c) 4-way multi-group

Figure 16: Group composition topologies used in simulation



(a) 10 msgs per batch (b) 30 msgs per batch

Figure 17: Average worst-case message latency. The number of members shown in the graph were distributed equally among all groups in the system. Messages were generated in a batch, and were sent by the application immediately after the logical token was released at each member, ensuring the worst-case latency for each message. Inter-group messages were either broadcast to all groups or sent to one other group with equal probability.

and system throughput. Figures 17a and 17b show the average worst-case latency of messages in both a single group and the four group composition for two different message generation rates. We simulated the multi-group system with several different proportions of inter-group to local group messages. Notice that even when 100% of the messages in the four-group composition are sent to the inter-group (the same communication pattern as a single group), the group composition shows lower latency. When fewer than 100% of the messages are sent to other groups, the performance of the system improves dramatically: with 20% inter-group messages, the latency of a 36 member four-group system is comparable to the latency in a single group with only 12 members. Even with 50% inter-group messages, the latency of the group composition is roughly 65% of the single group message latency. Since many applications will likely exhibit this type of communication locality, group compositions can provide substantial performance benefits to large applications.

Figure 18 shows another way to view the data from the previous experiements. In this graph, we can see that as more groups are added to the system, the aggregrate number of messages that can be sent within the system increases almost multiplicatively. At the same time, the average latency for each message decreases. This is the result of composing groups using multiple LANs: since the size of each group decreases as groups are added to the system, communication resources within each group are shared among fewer members. Together, these results indicate that group composition can be a powerful tool for building large-scale distributed systems. They also show that the semantic guarantees provided by the inter-group routers can be realized without compromising the performance and scalability benefits of the group composition. We are currently conducting further simulation experiments to study the effects of different topologies, group sizes, and message lengths.

7 Conclusions

This paper examined the problem of building scalable, fault-tolerant distributed systems from collections of communicating process groups while maintaining well-defined end-to-end delivery semantics.

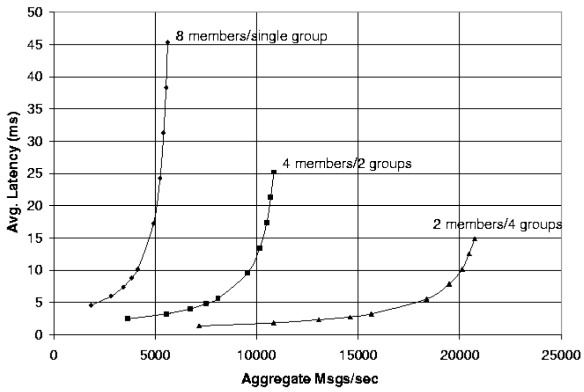


Figure 18: Aggregate throughput and worst-case latency for three basic system topologies. Each topology was run with members generating messages in batches as with figure 17. Batch sizes ranged from 1 to 30, with the batch size increasing from left to right along each curve. In all simulations members send 10% inter-group messages, 90% local messages. Inter-group messages are broadcast to all members or sent to one other group with equal probability.

Our approach is based on a new architecture for modularly composing process groups together to form larger distributed systems. We also presented a formal analysis of end-to-end message delivery semantics in systems composed of multiple process groups. Based on our analysis, a distributed system designer can determine what level of delivery semantic is required in each of the protocols in a composed system, and choose protocols which provide the required delivery semantics with minimal overhead. Finally, we presented results of a simulation study to evaluate the performance and scalability of the proposed architecture for a variety of inter-group topologies and communication patterns.

There are still a number of interesting unsolved problems related to group composition that require further work. The most important issue is probably adding support for timeliness guarantees on end-to-end messages. This is an active area of research, and we believe that the techniques being applied in other areas (such as QoS routing in the Internet) may also be applied here to achieve the same types of guarantees. Although providing real-time communication guarantees in a group composition is still a difficult and unsolved problem, there are techniques which would allow prioritized communication for inter-group messages while still enforcing semantic orderings between each message and all other messages of equal or lower priority class. We have designed a mechanism called *priority queues* which, when combined with the inter-group routers, can be used to implement prioritized communication. We are still in the process of evaluating this mechanism, but have achieved some encouraging results with it. We plan to continue to look at this and other means of achieving more predictable message delivery in group compositions. We also plan to implement the inter-group router architecture and test its scalabilty through experimental evaluation and simulation studies.

References

[1] Y. Amir, L. Moser, P. Melliar-Smith, D. Agarwal, and P. Ciarfella, "The Totem single-ring ordering and membership protocol," *ACM Transactions on Computer Systems*, vol. 13, no. 4, pp. 311–342, November 1995.

[2] K. Birman, A. Schiper, and P. Stephenson, "Lightweight causal and atomic group multicast," *ACM Transactions on Computer Systems*, vol. 9, no. 3, pp. 272–314, August 1991.

[3] R. van Renesse, K. Birman, and S. Maffeis, "Horus: A flexible group communication system," *Communications of the ACM*, vol. 39, no. 4, pp. 76–83, April 1996.

[4] S. Mishra, L. Peterson, and R. Schlichting, "Consul: A communication substrate for fault-tolerant distributed programs," *Distributed Systems Engineering*, vol. 1, pp. 87–103, 1993.

[5] S. Mishra, L. Peterson, and R. Schlichting, "Experience with modularity in Consul," *Software–Practice and Experience*, vol. 23, no. 10, pp. 1059–1075, October 1993.

[6] D. Powell, editor, *Delta-4 - A Generic Architecture for Dependable Dist. Computing*, Springer-Verlag, 1991.

[7] A. Tannenbaum, R. van Renesse, H. van Staveren, G. Sharp, S. Mullender, J. Jansen, and G. van Rossum, "Experiences with the Amoeba distributed operating system," *Communications of the ACM*, vol. 33, pp. 46–63, December 1990.

[8] D. Dolev and D. Malki, "The Transis approach to high availability cluster communication," *Communications of the ACM*, vol. 39, no. 4, pp. 64–70, April 1996.

[9] H. Kopetz and G. Grünsteidl, "TTP – a protocol for fault-tolerant real-time systems," *IEEE Computer*, vol. 27, no. 1, pp. 14–23, January 1994.

[10] T. Abdelzaher, A. Shaikh, F. Jahanian, and K. Shin, "RTCAST: Lightweight multicast for real-time process groups," in *Proceedings IEEE Real-Time Technology and Applications Symposium (RTAS '96)*, pp. 250–259, June 1996.

[11] P. Verissimo, P. Bond, A. Hilborne, L. Rodrigues, and D. Seaton, "The extra performance architecture (xpa)," in *Delta-4 - A Generic Architecture for Dependable Dist. Computing*, D. Powell, editor, Springer-Verlag, 1991.

[12] C. Fetzer and F. Cristian, "Fail-awareness: An approach to construct fail-safe applications," in *Proceedings of the International Conference on Fault-Tolerant Computing Systems (FTCS-27)*, pp. 282–291, Seattle, WA, June 1997.

[13] P. Verissimo and C. Almeida, "Quasi-synchronism: A step away from the traditional fault-tolerant real-time system model," *IEEE TCOS Bulletin*, vol. 7, no. 4, , December 1995.

[14] M. Hiltunen, X. Han, and R. Schlichting, "Real-time issues in Cactus," in *Proceedings of IEEE Workshop on Middleware for Distributed Real-time Systems and Services*, pp. 214–221, 1997.

[15] B. Whetten, "A reliable multicast protocol," in *Theory and Practice in Distributed Systems: Lecture Notes on Computer Science 938*, K. Birman, F. Mattern, and A. Schiper, editors, Springer-Verlag, 1995.

[16] Ö. Babaoğlu and A. Schiper, "On group communication in large-scale distributed systems," *ACM SIGOPS Operating Systems Review*, vol. 29, no. 1, pp. 612–621, January 1995. Also appears as Proceedings ACM SIGOPS European Workshop, September, 1994.

[17] Ö. Babaoğlu, A. Bartoli, and G. Dini, "Enriched view synchrony: A programming paradigm for partitionable asynchronous distributed systems," *IEEE Transactions on Computers*, vol. 46, no. 6, pp. 642–658, June 1997.

[18] L. Rodrigues and P. Verissimo, "Causal separators for large-scale multicast communication," in *Proceedings of the 15th International Conference on Distributed Computing Systems*, May 1995.

[19] D. Agarwal, L. Moser, P. Melliar-Smith, and R. Budhia, "A reliable ordered delivery protocol for interconnected local-area networks," in *International Conference on Networking Protocols*, 1995.

[20] R. Baldoni, R. Friedman, and R. Renesse, "Hierarchical daisy architecture for causal delivery," Technical report, Cornell University, 1996.

[21] S. Mullender, editor, *Distributed Systems*, chapter 5, pp. 104–107, in Mullender [28], second edition, 1993.

[22] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Communications of the ACM*, vol. 21, no. 7, pp. 558–565, 1978.

[23] S. Mullender, editor, *Distributed Systems*, chapter 5, p. 106, in Mullender [28], second edition, 1993.

[24] T. Ballardie, P. Francis, and J. Crowcroft, "Core based trees (CBT)," in *Proceedings of ACM SIGCOMM '93*, pp. 85–95, September 1993.

[25] F. Christian, "Synchronous atomic broadcast for redundant broadcast channels," *The Journal of Real-Time Systems*, vol. 2, no. 3, pp. 195–212, September 1990.

[26] K. Echtle and A. Masum, "A multiple bus broadcast protocol resilient to non-cooperative byzantine faults," in *Proceedings of Fault Tolerant Computing Symposium (FTCS) 26*, pp. 158–167, 1996.

[27] T. Abdelzaher, A. Shaikh, S. Johnson, F. Jahanian, and K. Shin, "RTCAST: Lightweight multicast for real-time process groups," Technical report, Dept. of Electrical Engineering and Computer Science, University of Michigan, 1997. To appear in IEEE Transactions on Software Engineering.

[28] S. Mullender, editor, *Distributed Systems*, Addison-Wesley, second edition, 1993.

Appendix A: Proofs

Proof of lemma 3:

As stated in the lemma, we assume that the causal delivery properties are enforced by each group's delivery protocol. We now show that each of these properties is enforced on messages sent between the process groups.

Proof of property 1 follows directly from lemma 2, since it is simply sender-based FIFO delivery. To prove that property 2 is enforced for messages sent between groups, assume without loss of generality that a message m_i is sent from a process in group A to a set of receivers in both groups, and m_i is dependent on a causally preceding message m_h. For receivers in group A, if m_h was also sent by a process in group A, then causal ordering of the two messages will be enforced by A's delivery protocol. If m_h was sent by a process in group B, then from group A's perspective m_h was sent by the inter-group router. to This would generate the sequence of events recv$(m_h) \rightarrow$ send(m_i) at the sender of m_i. By definition, therefore, group A's causal delivery protocol will deliver m_i after m_h at all receivers in group A.

For receivers of m_i in group B, there are two possible cases: m_h was sent by a process in A, or it was sent by a process in B. If m_h was originally sent by a process in A, then because of A's causal delivery protocol, m_i must have been delivered after m_h in A, and because of the FIFO forwarding mechanism must have been sent after m_h to the inter-group protocol. Since the inter-group protocol enforces sender-based FIFO, m_h will also be forwarded to group B before message m_i, and will therefore be delivered before m_i to processes in B, since the inter-group router is the same sender for both in B. If m_h was sent by a process in group B, then within group B the inter-group router generated the sequence of events recv$(m_h) \rightarrow$ send(m_i), and m_i will be delivered after m_h in group B by B's causal delivery protocol. This argument can be extended iteratively to show that property 3 is also enforced. $\Box$

Proof of lemma 5:

From the description of the inter-group router architecture, we know that messages are delivered to the inter-group router of group g_s with the semantic S_{g_s}. The inter-group router sends the messages to the inter-group protocol in the same order, and because the inter-group protocol enforces sender-based FIFO, the messages will be delivered to the inter-group routers of all receiving groups in the same order. Finally, since each receiving process group supports sender-based FIFO, the messages will be delivered to all processes in the same order. Therefore, all messages from g_s will be delivered at all receivers with semantic S_{g_s}. $\Box$

Proof of lemma 7:

To prove this lemma we shall show that the requirements for causal delivery given in table 3.1 are enforced by the process group and inter-group protocols.

As with the paired group case, property 1 is implicitly enforced by the FIFO forwarding mechanism of the inter-group routers. Since causal delivery is inherently sender-based FIFO, and the receiving groups also support sender-based FIFO, messages from a single sender are always delivered in the order they were sent.

To show that the other properties are also enforced in this composition, consider a message m_i which is sent by a process in some group $g_i \in G_S$. When m_i is delivered to receiving processes within g_i, there are two possible causal dependencies. It could be dependent on a message m_h which was previously sent by another process within g_i. In this case, g_i's group communication protocol will ensure that both messages are delivered in an order that respects their causal relationship, since $g_i \in G_S$ and all groups in G_S enforce causal delivery semantics. If m_i is causally dependent on some message m_h which was sent to g_i by another process group g_h, then the sender of m_i must have received m_h prior to sending m_i (property 2 in table 3.1). In this case g_i's delivery protocol will see the sequence of events recv(m_h) $\rightarrow$ send(m_i) at the sender of m_i, and will deliver m_h before m_i, preserving causality.

When m_i is delivered to a receiver in another process group g_r, there are again two possible ways that it could be causally dependent on other messages. If m_i is causally dependent on some message m_h that was sent from either g_i or g_r, then this is the same as the paired group composition case, and by lemma 3 m_i will be delivered after m_h. If, however, m_i is causally dependent on some message m_h that was sent by another group $g_k \in G_S$, then a process in g_i must have received m_h before sending m_i. In this case, the inter-group protocol will observe the events recv(m_h) $\rightarrow$ send(m_i) at the inter-group router for g_i. Since the inter-group protocol enforces causal delivery, it will delivery m_i after m_h at all receivers of m_i and m_h, preserving the causal ordering between them.

The preceding cases can be applied symmetrically to every message receive event for every message in M. Therefore, this composition will deliver all messages to all receivers in an order that respects the causal precedence relationships of the messages in M. $\square$

Proof of lemma 8:

Since each sending group enforces total delivery, all messages will be delivered in a total order to all receivers in the group, including the inter-group router. Inter-group messages are then delivered in the inter-group protocol in a total order at all inter-group routers. By definition, this order will also be sender-based FIFO, so the existing total order of each sending group's messages is preserved. Finally, messages are delivered in sender-based FIFO order to all receivers. This will preserve the message ordering that was enforced by the inter-group protocol, so all inter-group routers will forward all inter-group messages in total order, and all inter-group messages will be delivered to all receivers in total order. $\square$

Proof of lemma 9:

To prove this lemma, we split inter-group message delivery into two phases. In the first phase, each message is delivered by the sender's group protool to an inter-group router, which sends it in the inter-group protocol. In this phase, there is only one receiver in the group: the inter-group router. Since there is only one receiver, any order of messages received by the inter-group router constitutes a total order. In the second phase, the inter-group protocol delivers all messages to the inter-group routers in a total order. The inter-group routers then forward the messages in the same total order to all receivers in the attached process groups. Therefore, each receiver will receive the same total order of messages. $\square$

Proof of lemma 10:

To prove this, consider three possible delivery semantics for S. For S = sender-FIFO, the proof

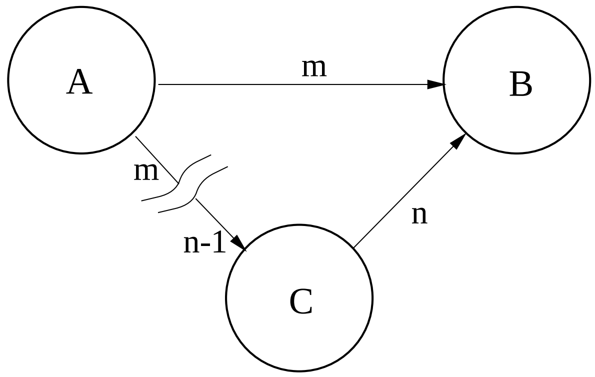


Figure 19: Example of an incorrect causal ordering

follows directly from the definition of FIFO and lemma 5. For both causal and total ordering, note that by definition, messages will be delivered to other processes in the sender's group with delivery semantic S, since S is enforced by the local group protocol. To show that the same semantic holds when messages are also delivered to processes in the receiving group, there are two cases to consider. First, consider the case where the receiving group is not one of the sending groups (i.e. $g_r \notin G_S$). If S = causal delivery, then the only way a message can be causally dependent on another is if both originated in the same process group. In this case, they will be delivered to the inter-group router in causal order by the group protocol, and that order will be preserved by FIFO forwarding until the messages are delivered at all receivers. If S = total ordering, then messages will be delivered in the sending group in total order, and the inter-group router will forward messages in the same order. Since the inter-group protocol and the receiving group protocol are sender-based FIFO, they will also preserve that ordering until the messages are delivered to all processes in the receiving group. Messages that originate in different process groups will be delivered in an arbitrary order by the inter-group protocol to the receiving group's inter-group router, but since it is the only receiver in the inter-group protocol, that ordering by definition constitutes the total ordering of the messages. That ordering is then preserved by the FIFO forwarding mechanism until the messages are delivered to the receivers.

If $g_r \in G_S$, the same arguments apply. The only difference is that messages sent within g_r may be semantically dependent on messages that arrived from other groups. First, we consider the case where S is causal delivery. We have just shown that all messages arriving to g_r will be delivered to receivers in the g_r in causal order. The delivery protocol of g_r will ensure that those messages are delivered in causal order at all receivers with respect to messages sent entirely with g_r. If S is totally ordered delivery, we have already shown that messages from other groups will be delivered to receivers in g_r in total order. In addition, g_r's communication protocol will totally order messages arriving from the inter-group network with messages sent only within g_r. There are no other groups in the system receiving messages from g_r, so all messages in the M will be delivered to all receivers in total order. □

Proof of lemma 11:

By lemma 5, messages from each group are delivered to all receivers in causal order, since each group enforces causal delivery and the inter-group protocol enforces sender-based FIFO. Therefore, the only way a causal ordering violation can occur is among messages from different senders.

Assume that in some implicitly causal pipeline there is a group, B, which receives two messages, m

and n, each from a difference sending group (see figure 19). Assume without loss of generality that message n is causally dependent on message m, but B incorrectly delivers n before m. Suppose n was sent by some group, C, as shown in the figure. Since n is causally dependent on m, by definition C must have received either m or a message $n-1$ which was also causally dependent on m. Let us call the sender of m group A.

If C and B both received m, then A must communicate directly with both B and C. We already know that there is a directed path from C to B, so this means there is also a directed path ($A \rightarrow\ C \rightarrow\ B$) from A to B. Since A communicates directly with B, this is a contradiction of our original assumption that B was in an implicitly causal pipeline.

If C didn't receive m, but instead received a causally dependent message $n-1$, the sender of $n-1$ must have originally received m or another message $n-2$ which was causally dependent on m. Following the chain of causality, we must eventually reach a node that received m directly from A, implying that there is a directed path from A to C, and thus from A to B. Once again this is a contradiction of our original assumption that B was in a causal pipeline. $\square$

Proof of lemma 12:

By lemma 5, messages from each process group must be delivered in total order at all receivers. Therefore, the only way a total ordering violation can occur is if two groups deliver two messages from two different sending groups in a different order. Since by definition no two groups in an implicitly totally ordered pipeline can ever receive messages from the same two senders, this condition can never occur, and the totally ordered pipeline implicitly guarantees that total ordering will be observed on all messages at all receivers. $\square$