

The Inter-group Router Approach to Scalable Group Composition *

Scott Johnson, Farnam Jahanian, and Jigney Shah
Dept. of Electrical Engineering and Computer Science
University of Michigan
1301 Beal Ave.
Ann Arbor, MI 48109-2122
{scottdj,farnam,jiggy}@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. We propose a new architecture which supports modular group composition by providing a distinction between intra-group and inter-group communication. With this architecture, multiple group communication protocols and end-to-end delivery semantics can be used in a single system. These features reduce the complexity of ordering messages in a group composition, resulting in enhanced scalability. Finally, we present simulation results comparing the performance of a group composition using our architecture to that of a single process group.

1. Introduction

Group communication is a widely studied paradigm for building distributed systems. In this approach, a distributed application is structured as a group 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 computational load, or to support rapid dissemination of information from senders to a group of potentially anonymous recipients.

It has been recognized that traditional group communication techniques may not scale well when used with large scale distributed applications. For example, a number of group communication protocols make use of a logical token

ring [2, 15, 7]. In these protocols, message latency typically grows linearly with the size of the group and the number of messages, since each new process proportionally increases the time it takes for the token to rotate. In other protocols, as the number of processes in a group increases messages often incur higher overhead due to contention for some limited group resource. Eventually, a limit is reached beyond which additional processes cause group performance to become unacceptably slow. These factors can pose an undesirable limit to the scalability of distributed applications built using a single process group.

Researchers have primarily taken one of two approaches to improve group communication scalability. Some have attempted to improve the scalability of the single process group model [4], and its use in wide-area environments [3, 17] even in the presence of network partitions [8]. Others have designed architectures which allow systems to be composed from multiple process groups while still maintaining some type of delivery semantics on messages sent between groups [15, 5, 10].

In our experience, many distributed applications have an inherent organization which can easily take advantage of the group composition approach. These applications can be broken down into subsets of functionally related processes which communicate much more frequently with each other than with other processes in the system. In this case, each subset of processes can be composed as a separate process group. This allows standard methods of replication and consistency management to be performed independently for each group, without the resource contention and extra overhead that would be incurred if all processes were placed in a single group. In addition, each group can be located on a separate physical network so that it can use the full bandwidth of the medium for its internal messages.

In this paper, we propose a novel architecture which supports the modular composition of process groups, while maintaining well-defined, flexible end-to-end delivery semantics. It provides a distinction between intra-group

*This work is supported in part by a research grant from the Defense Advanced Research Projects Agency, monitored by the U.S. Air Force Rome Laboratory under Grant F30602-95-1-0044.

and inter-group communication, allowing each group to use a different communication protocol to more efficiently implement the application’s communication requirements. Each group operates autonomously, minimizing shared state maintenance across groups and improving scalability over systems that require groups to actively cooperate to deliver messages. Our architecture also supports complex composition topologies, allowing a distributed system to be organized efficiently based solely on the application’s communication requirements.

The remainder of this paper is organized as follows: Section 2 describes previous work in this area and compares our approach to existing group composition architectures. Section 3 describes our distributed system model and group composition architecture. Section 4 provides an analysis of end-to-end delivery semantics for various composition topologies, and demonstrates how our architecture can provide specific delivery guarantees. Finally, section 5 presents simulation results to validate our claims of scalability.

2. Related Work

Historically, most group communication protocols have focused on providing fault-tolerant message delivery in asynchronous systems, i.e. systems with no lower bound on the communication time between processes. These protocols support various delivery semantics such as FIFO, causal, or total ordering while providing a consistent membership view of participants believed to be accessible. Examples include Totem [15], Horus [18], Consul [14], Delta-4 [20], and Transis [8]. A number of systems also provide explicit support for real-time message delivery, such as TTP [13], RTCAST [1] and XPA [20]. Other approaches are based on the best effort paradigm, including the *fail-awareness* framework [9] and the *quasi-synchronous* model [19]. It is the goal of our group composition architecture to provide a framework for composing these protocols together to form heterogeneous systems that efficiently implement the communication requirements of a distributed application.

In the introduction, we listed several architectures which support group composition. Each was designed to compose groups into a single ordering domain [6] which enforced a particular end-to-end delivery semantic on messages exchanged by the composed groups. Both the totem multi-ring architecture [15] and the approach taken by Fritzke, et al [10], compose process groups which all use a single group communication protocol, and enforce total delivery across all messages sent between groups. In these architectures, all processes in the destination groups for each message must interact with each other to agree on a delivery order. The hierarchical daisy architecture [5] enforces end-to-end causal delivery among composed process groups. It separates intra-group from inter-group communication, allow-

ing messages to be delivered independently in each composed group. It also propagates message delivery information throughout the system so that messages do not need to be buffered after they have been delivered at all destinations.

In this paper, we propose a general solution for scalable group composition that does not depend on a particular communication protocol or delivery semantic. It separates intra-group from inter-group communication, and only one process from each group is required to interact with other processes to reach a consensus on message delivery order. In addition, our architecture considers the enforcement of multiple ordering domains within a single composed system. For example, although global total delivery is a very strong guarantee, it has been observed that there are few systems which require such a strong delivery semantic among all groups[6]. By utilizing multiple ordering domains, our architecture can improve the scalability of group compositions by enforcing a particular delivery semantic only among groups that require it. This also allows the construction of much more complex composition topologies compared to existing single-domain architectures.

There is an important difference between the system model of the Totem multiple ring protocol and our approach which bears further discussion. The Totem multiple ring protocol organizes the system as a collection of interconnected local-area networks, called rings. Each group in the system may contain processes which are located on different rings, and each process may be a member of multiple groups. This allows much greater flexibility in the allocation of processes to groups, but it incurs a much higher cost to order messages for each group when its members are not located in the same part of the network. Our architecture requires that processes which belong to the same group be colocated in the network. This enables us to support modular group composition and more flexible delivery semantics, and it reduces the amount of global state information that must be propagated between groups.

3. Group Composition

We assume that a composed system consists of a set of processes $P = \{p_1, p_2, \dots, p_n\}$, which are organized into one or more process groups $G = \{g_1, g_2, \dots, g_m\}$, such that each process is a member of exactly one group¹ (i.e. groups are non-overlapping). The statement $p_i \in g_j$ denotes that process p_i is a member of group g_j , and we say that g_j is p_i ’s *local* group. All other groups are considered to be *remote* groups with respect to p_i . Groups can be *open* (messages can be sent to the group by non-member processes) or *closed* (messages can only be sent to a group by members of that group). We make no assumptions about whether the system is synchronous or asynchronous, since

¹A process can be a member of more than one group if each one is in a separate ordering domain. Ordering domains are defined in Section 4.

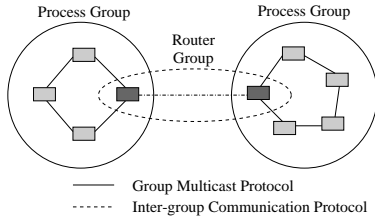


Figure 1. A simple group composition. Two inter-group routers connect two groups via an inter-group communication protocol.

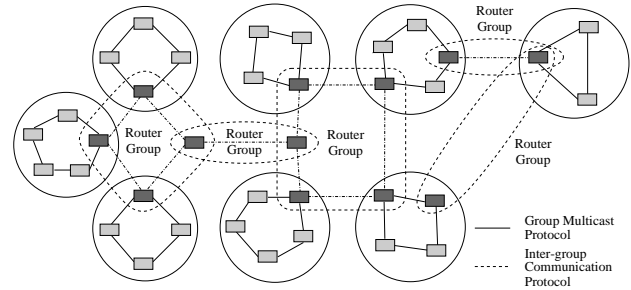


Figure 2. A more complicated composition, using multiple inter-group communication protocols and multiple routers.

our architecture can be used in systems of either type. We also make no assumptions about the assignment of physical networks to process groups. Each group can be composed on a separate physical network (the ideal solution), or multiple groups can coexist on the same network if their protocols allow it.

When describing the operation of a group communication protocol, we make a distinction between a process receiving a message, and delivering that message. By *receive*, we mean that the message has been received from the network by the group communication service, but has not yet been seen by the application. At some later time, the group communication service *delivers* the message to the application process, usually after it has determined that it can do so without violating any guaranteed delivery semantics.

Communication in our model is divided into two types: *intra-group* and *inter-group*. In intra-group communication, an intra-group message is sent by some process $p_i \in g_j$, and is destined only for other processes in g_j . In inter-group communication, an inter-group message is sent by some process $p_i \in g_j$ and is destined for any subset of groups $G' \subseteq G$, such that $G' \neq \{g_j\}$ (i.e. the message is not an intra-group message). Messages sent to a group are delivered to all members of that group². Note that the sender's group can be included in the destination list of an inter-group message, provided the message is sent to at least one additional group. When we discuss the end-to-end delivery semantics of messages in a group composition, we mean the delivery semantics observed over both intra-group and inter-group messages. In other words, a global delivery semantic.

3.1. Realizing Inter-group Communication

In our group composition architecture, intra-group communication is unchanged from the single-group model. Processes use their local group communication protocol to send intra-group messages to other members of their group.

²This assumption is not actually required by our architecture, but it simplifies the discussion of inter-group message delivery.

In order to realize inter-group communication, we introduce the notion of an *inter-group router*. An inter-group router is a special process which is capable of forwarding messages between two or more communication protocols. At least one inter-group router is added to each process group in the system. It joins that group as an application-level process, and to the group communication protocol it is indistinguishable from any other member of the group. Inter-group routers communicate with each other using one or more inter-group communication protocols, which are external to the process groups. Each inter-group communication protocol can be as simple as an unreliable unicast (such as UDP), or as complex as a group multicast protocol. In the latter case, we refer to the collection of inter-group routers connected by that protocol as a *router group*. It is even possible that an inter-group communication protocol operates across a wide-area network, connecting process groups which are geographically distant.

Figure 1 shows a simple composition which uses inter-group routers to compose two process groups. A more complex composition is given in Figure 2, which demonstrates how groups can be connected by multiple inter-group routers and inter-group communication protocols. Note that in this composition, an inter-group router is used to connect two inter-group communication protocols. This is consistent with our definition of an inter-group router as a process which can forward messages between two or more communication protocols. Also note that it is possible to connect more than two protocols using a single inter-group router, as shown by the rightmost router in the figure.

This architecture is reminiscent of traditional network routing architectures seen in the Internet, in which autonomous systems make use of an *interior routing protocol* to perform routing within their internal network, and an *exterior routing protocol* to exchange routing information with other autonomous systems. Usually, some type of border router or gateway serves as the interface between the

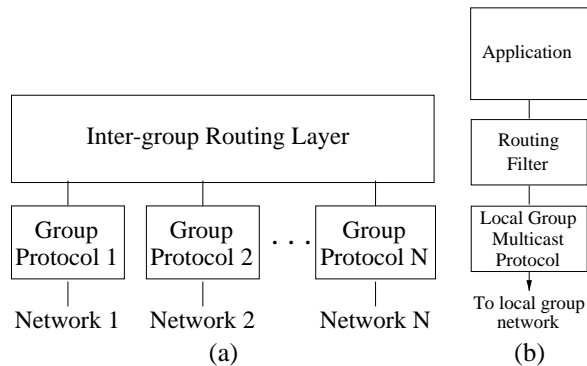


Figure 3. Protocol stack for (a) inter-group routers and (b) group members

two types of protocols [11], which is analogous to the inter-group router in our architecture.

Because of the potential complexity of the composition topology, there may be more than one communication path between groups in a composition. In this case, the inter-group routers must be able to select an appropriate next hop when forwarding messages. This can be accomplished either through statically configured routes or by using a dynamic routing protocol. Finally, since the inter-group routers form a single point of failure in the system, standard active or passive replication techniques can be used to improve fault-tolerance. Due to space constraints we do not present further details of these features here.

3.2. Inter-group Router Architecture

Figure 3(a) shows the software architecture of an inter-group router connecting N communication protocols. It consists of the inter-group routing layer and the protocol stack for each group to which it is connected. If a group is composed on a separate physical network, the inter-group router will also have a separate network interface for that group. Each application process must also have a small filter inserted in its protocol stack between the application and the group communication protocol, as shown in Figure 3(b). This filter exports a multi-group addressing scheme to the application, and ensures that inter-group messages sent by the process are delivered to the local inter-group routers by the local group communication protocol. If necessary, this filter can also insert other header information to help the inter-group routers forward messages to their destinations.

When an inter-group router forwards messages between communication protocols, it does so in FIFO order. In other words, inter-group routers do not reorder inter-group messages when forwarding them. This guarantees an interesting property on inter-group communication, as shown in the following theorem. As we will show in section 4, this prop-

erty is crucial for guaranteeing end-to-end delivery semantics in group compositions.

Theorem 1 (FIFO Forwarding) Given a process group G which is part of a group composition, and a set of inter-group messages M which are sent by processes in G , assume that G delivers messages according to some ordering semantic S . If all communication protocols in the composition enforce sender-based FIFO³ delivery, and all messages in M are forwarded through the same sequence of inter-group routers for each destination, then all messages in M are delivered with ordering semantic S to all receivers.

Proof: Proof of this theorem follows directly from the definition of sender-based FIFO and the fact that inter-group routers do not reorder messages when forwarding them. \square

3.3. Inter-group Message Forwarding

We now describe how inter-group messages are delivered to their destinations by the inter-group routers. When an application process sends a message, the inter-group routing filter at that process determines whether the message is destined for the local group only (i.e. an intra-group message), or is also being sent to another group (i.e. an inter-group message). For intra-group messages, the routing filter simply passes the message through unchanged, and it is delivered normally by the local group communication protocol. For inter-group messages, the routing filter inserts a header in the message indicating the destination groups' addresses and any other information needed by the inter-group routing protocol. This header information may include a unique message id which enables the inter-group routers to detect duplicate messages. The message is then delivered to an inter-group router in the local group using the local group multicast protocol. Unless it is an inter-group message requiring total ordering, the message is also delivered to receivers in the local group at the same time, using that group's standard group multicast operation. The reason totally-ordered inter-group messages cannot be delivered immediately is explained in Section 4.

Based on the routing header, each inter-group router that receives a message forwards it to the appropriate next-hop router until it is delivered to an inter-group router in every destination group. Those routers then deliver it to the destination processes using their local group multicast protocol. The routing filter at each receiving node strips off the routing header and delivers the message to the application. To the communication protocol in each group, the message is indistinguishable from a normal intra-group message.

4. End-to-End Delivery Semantics

In the previous section, we described how process groups could be composed together using inter-group routers. This

³Readers not familiar with sender-based FIFO can find a definition in [16].

solved half of the problem: process groups in a composed system could exchange messages, but there were still no guarantees on end-to-end message delivery. In this section we examine the problem of enforcing end-to-end delivery semantics in composed systems. We will show that it is possible to compose a system using inter-group routers and appropriately chosen inter-group communication protocols such that end-to-end application delivery requirements are met. Before we begin our analysis, however, we must define several concepts which will be important to our discussion.

4.1. Ordering Domains

Our analysis of end-to-end delivery semantics is based on the notion of *ordering domains*, described by Birman in [6]. An ordering domain consists of a subset of groups in a distributed application which exchange messages and require a particular delivery semantic to be enforced on those messages. More formally, an ordering domain consists of a set of groups $D \subseteq G$, and an associated delivery semantic S . If M is the set of all inter-group messages sent by processes in D and whose destinations include groups in D , then all messages in M are delivered in an order consistent with the delivery semantic S to all receivers in D . Each distributed application consists of at least one and possibly many ordering domains. In the most extreme case, a distributed application which has no end-to-end delivery requirements will have m ordering domains, one for each group in the system. Note that this definition of ordering domains is slightly more general than Birman’s, allowing each group to be part of multiple ordering domains.

By dividing the application into ordering domains, a message ordering problem can be broken down into several smaller, more manageable sub-problems which can be solved independently. For example, if group g_1 is in one ordering domain and group g_2 is in another, then messages from g_1 do not have to be ordered with respect to messages from g_2 .

There are a number of delivery semantics that may be enforced in a single ordering domain. In our analysis we focus on the following: sender-based FIFO, causal, and total delivery. Due to space limitations we do not define these semantics here, but formal definitions are available in [16]. In addition, we will also use FIFO total delivery, which simply enforces both sender-based FIFO and total order.

We will first show how to compose groups using inter-group routers to implement a single ordering domain. We will then show how multiple ordering domains can be connected to implement a complete distributed system.

4.2. Single Ordering Domain Compositions

Paired Group Composition: We begin our analysis by examining the smallest ordering domain possible: one consisting of only two process groups, which we call a *paired group* composition (Figure 1). This composition can also

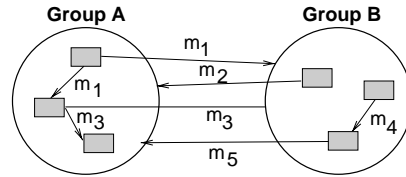


Figure 4. Example of causal delivery in a paired group composition. Shaded blocks represent processes and directed edges represent application-level communication. Inter-group routers are not shown.

be viewed as a *client-server* group composition, and is a commonly used architecture in distributed systems. In this composition, two process groups communicate via a single inter-group communication protocol. Although this is a simple composition, it illustrates many of the basic issues affecting delivery semantics in a composed system. As we will show, it is possible to provide strong end-to-end delivery semantics in this composition even when using lightweight inter-group communication protocols. First, let us consider a specific communication scenario:

Example: Consider a paired group composition in which both groups enforce causal delivery, and the inter-group communication protocol is sender-based FIFO (Figure 4). A process in group A sends message m_1 to both groups. Concurrently, a process in group B sends message m_2 to group A. A process in group A receives m_2 , and then sends m_3 to both groups. A process in group B receives m_3 and sends m_4 to group B only. Finally, a process in group B receives m_4 and sends message m_5 to group A. \diamond

Given the above scenario, m_1 and m_2 can be delivered in any order with respect to each other since they are concurrent. Within group A, m_3 will be delivered after m_1 , since A enforces causal delivery and m_3 was sent by a process in A after receiving m_1 . Within group B, m_3 will be delivered after m_1 since B enforces causal delivery and a member of B (the inter-group router) sends m_3 after sending m_1 . Similarly, m_3 will be delivered after m_2 , since the inter-group router in B sends m_3 after receiving m_2 . Next, m_4 is delivered after m_3 at all receivers within group B, since m_4 was sent by a member of B after it had received m_3 . Finally, within group A, m_5 is sent by the inter-group router after it received causally preceding message m_3 , and so m_5 will be delivered after m_3 to all members of A, even though A never saw m_4 . Thus, all messages in this example are delivered in a global causal order at all receivers in both groups.

This example demonstrates how inter-group routers provide an abstraction to each process group in the composed system, making all other groups appear to be a single process which is a local member of the group. It is this abstrac-

Semantics of:			Effective End-to-End
Sender	Inter-Group	Recv	
Any	FIFO	Any	FIFO
Causal	FIFO	Causal	Causal
Total	Total For.	FIFO Total	Total

Table 1. End-to-End Delivery Semantics in Paired Group Compositions

tion which enables process groups to be composed modularly, allowing them to exchange messages without having to interact directly and without losing causal precedence information. We will show later that this same guarantee can be enforced for total delivery as well. In general, we can enforce global causal delivery regardless of the actual sequence of messages, as shown in the following theorem:

Theorem 2 (Paired Causal Delivery) Given a paired group composition, assume processes in both groups send any combination of intra-group and inter-group messages. If each group enforces causal delivery, and the inter-group protocol enforces sender-based FIFO delivery, then all messages will be delivered in causal order at all receivers. \square

Due to space constraints, we are unable to include formal proofs here. The reader is referred to our technical report [12] for formal proofs of theorems appearing in this paper.

Informally, this property can be seen to hold based on the behavior observed in the example given above. The inter-group router in each process group acts as a local sender and receiver of inter-group messages, allowing each group protocol to see the causal relationships between all messages it delivers.

Unfortunately, total ordering cannot be enforced in the same way. Consider the following scenario: Assume two groups in a paired composition enforce total delivery. A process in group A sends inter-group message m_1 to both groups at the same time that a process in group B sends message m_2 to both groups. Message m_1 may be delivered in group A and forwarded by the inter-group router to group B at the same time that message m_2 is delivered in group B and forwarded to group A. Since m_1 has already been delivered in A before m_2 is sent in A by the inter-group router, processes in A will deliver m_2 second. For the same reason, m_2 may be delivered first in group B, violating total order. Even if the inter-group communication protocol enforces total ordering, it cannot prevent this situation, since each group delivers locally originated messages first.

To ensure an end-to-end total order, there must be a single entity responsible for ordering all messages. To achieve this, messages must initially be sent to the inter-group router *only*, and not delivered to the rest of the sender’s group. The

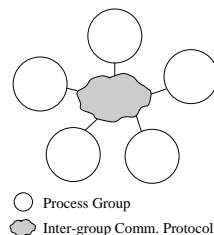


Figure 5. Sample multi-group composition.

inter-group router will then send them to the inter-group communication protocol, which must totally order the messages and deliver them back to the inter-group routers. Only then can the inter-group routers deliver the messages to their local group with no chance of a conflict in another group. We will refer to this method of forwarding messages as *totally ordered message forwarding*, or simply total forwarding. Furthermore, in order for the delivery order enforced by the inter-group communication protocol to be preserved, each process group must enforce FIFO total delivery.

Although requiring messages to be ordered by a single protocol may seem to be a limitation of our architecture, this is a problem that would be experienced in any group composition system [6]. It is still cheaper than using a single process group, since for inter-group messages total ordering only needs to be enforced among two processes (the inter-group routers), not every process in the system. It is important to note that very few distributed applications actually require end-to-end total ordering.

The following theorem summarizes these properties:

Theorem 3 (Paired Total Delivery) Given a paired group composition in which the inter-group routers perform totally ordered message forwarding and enforce total delivery, if each group enforces FIFO total ordering, then all messages will be delivered at all receivers in total order. \square

Proof: Because of total forwarding all messages are delivered to the inter-group routers in total order. By theorem 1, and since both groups support FIFO total delivery, all messages will be delivered in the same total order at all receivers in both groups. \square

A summary of the delivery semantics of the paired group composition is given in Table 1.

Multi-Group Composition: Like the paired group composition, the multi-group composition utilizes a single inter-group communication protocol. However, it expands on the paired composition by composing additional process groups to realize larger ordering domains. An example is shown in Figure 5(a). This composition is equivalent to a single daisy from the Causal daisy architecture [5], Totem Multiple-ring Protocol’s “n-plus-1 ring” composition [15], and the architecture of the fault-tolerant total order multicast described

Semantics of:			Effective End-to-End
Sending	Inter-Group	Recv	
Any	FIFO	Any	FIFO
Causal	Causal	Causal	Causal
Total	Total For.	FIFO Total	Total
All other cases			FIFO

Table 2. Basic End-to-End Delivery Semantics in a Multi-group Composition

by Fritzke, et al, in [10]. Note that the paired-group composition is really just a special case of the multi-group composition. We now extend the results of the paired-group semantic analysis to the multi-group composition. We consider two patterns of communication: one-to-many, and many-to-many.

One-to-many Communication: In one-to-many communication, a single group sends inter-group messages to one or more receiving groups in a multi-group composition. By theorem 1, as long as each communication protocol enforces sender-based FIFO ordering, all messages will be delivered to all receivers with the sending group’s delivery semantic, since there is a single communication path between each pair of groups. Though simple, this type of communication can be very useful when the results of a single process group need to be published to multiple groups.

Many-to-many Communication: In many-to-many communication, two or more process groups send messages to two or more receiving groups. Since there are now multiple senders and receivers in the inter-group communication protocol, simple FIFO delivery is no longer sufficient to guarantee an end-to-end delivery semantic. However, as the following theorem shows, if the inter-group communication protocol also enforces causal delivery, causal end-to-end delivery can still be enforced:

Theorem 4 (Multiple Sender Causal Delivery) Consider a set of messages M that 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. If every group in G_S enforces causal delivery, every group in G_R enforces at least sender-based FIFO delivery, and the inter-group communication protocol enforces causal delivery, then all messages in M will be delivered to all receivers in causal order. \square

For end-to-end total delivery, there are two cases to consider: when the set of receiving groups does not overlap with the sending groups, and the general case where any group may send or receive inter-group messages. The following theorems describe these configurations. Proofs of

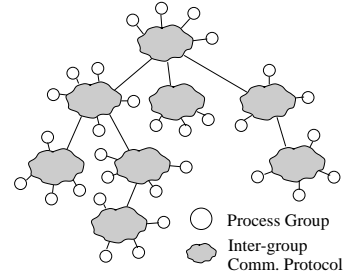


Figure 6. Hierarchical multi-group composition. For simplicity, the inter-group routers are not shown.

these theorems are similar to those for the paired group composition, and can be found in [12].

Theorem 5 (Non-overlapping Total Delivery) Consider a set of messages M which are sent within a multi-group composition by some set of sending groups G_S to a set of receiving groups G_R , such that $|G_S \cap G_R| = 1$. If all groups in G_S enforce total delivery, and both the groups in G_R and the inter-group communication protocol enforce FIFO total delivery, then all messages in M will be delivered in total order to all receivers. \square

Informally, since there is at most one group which both sends and receives inter-group messages, the problem of simultaneous sends can not occur. In this case, using total forwarding is not necessary, since the FIFO total inter-group protocol will be sufficient to ensure totally ordered message delivery among all inter-group routers.

In the more general case where two or more process groups send totally ordered messages to each other, simultaneous senders may again cause total delivery to be violated. To solve this problem, we again use total forwarding to force all messages to be ordered by the inter-group communication protocol.

Theorem 6 (Multi-Group Total Delivery) Consider a set of messages M which are sent within a multi-group composition. If the inter-group routers perform total forwarding, and all communication protocols enforce FIFO total delivery, then all messages in M will be delivered in total order at all receivers. \square

Once again, it is rare that an application would require such a strong delivery guarantee. In any case, enforcing this semantic in multiple smaller groups should be cheaper than trying to enforce it in a single large process group. A summary of many-to-many communication semantics is given in Table 2.

Hierarchical Multi-Group Composition: In some cases, an application may require more groups to be composed as a single ordering domain than can reasonably be

supported by the multi-group composition’s single inter-group communication protocol. We now describe a class of compositions, called *hierarchical multi-group compositions*, which allows several multi-group compositions to be connected together to form a larger ordering domain.

A hierarchical multi-group composition can be formed by taking any number of multi-group compositions and connecting their inter-group communication protocols using inter-group routers. The resulting topology must form an (undirected) acyclic graph. One of the multi-group compositions is then selected as a root, and the entire topology is viewed as a tree rooted at that composition. Since the composition is a tree, by definition there is a unique path between any two groups in the system. Multi-group compositions which communicate with each other frequently should be placed as close to each other in the tree as possible, to minimize the communication path between them. Similarly, end-to-end communication in the system will be faster if the tree is balanced. A sample hierarchical multi-group composition is shown in Figure 6.

By theorem 1, sender-based FIFO delivery can be trivially enforced as before, with each inter-group communication protocol forwarding messages between groups using sender-based FIFO delivery. For causal delivery, consider the following theorem:

Theorem 7 (Hierarchical Causal Delivery) Given a hierarchical multi-group composition, if every process group and inter-group communication protocol enforces causal delivery, then all messages in the composition are delivered in causal order at all receivers. □

Informally, note that by definition messages destined for the same destination group must be delivered to the same inter-group router at least once during their traversal of the tree. By repeated application of theorem 4, it can be shown that all messages sharing a destination group will be causally ordered with each other by the time they reach their destination.

In order to enforce end-to-end total delivery, we must introduce a variation of the total forwarding algorithm from above, called *hierarchical total forwarding*. Hierarchical total forwarding is an extension to total forwarding, such that whenever an inter-group message is sent to a set of receiving groups G_r , it is delivered first in the inter-group communication protocol that is the least-common ancestor of the groups in G_r (in other words, the root of the smallest sub-tree that includes all groups in G_r). With this forwarding mechanism, we can enforce end-to-end total delivery as shown by the following theorem:

Theorem 8 Given a hierarchical multi-group composition, if every process group and inter-group communication protocol enforces FIFO total order, and hierarchical total forwarding is used to deliver messages in the inter-group communication protocols, then all messages will be delivered in total order at all receivers. □

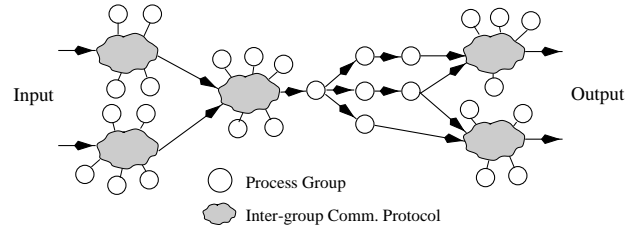


Figure 8. Example of a pipeline using multiple ordering domains.

warding is used to deliver messages in the inter-group communication protocols, then all messages will be delivered in total order at all receivers. □

Informally, we can see that every message delivered to a given set of receivers will be totally ordered with respect to all other messages sent to those receivers, by definition of hierarchical total forwarding.

In this section, we have so far shown how to compose process groups into a single ordering domain. We now examine the problem of connecting all of a distributed application’s ordering domains to form a complete system.

4.3. Multiple Ordering Domain Compositions

Using the paired, multi-group, and hierarchical multi-group compositions, distributed applications can compose process groups into a single ordering domain which enforces a desired end-to-end delivery semantic. However, these compositions are still limited by the scalability of the inter-group communication protocols. This is a problem that would be experienced by any group composition architecture given the same number of inter-group messages. Fortunately, many applications do not require a single ordering semantic to be enforced over all messages. They can instead be broken down into multiple ordering domains, and compose each domain using one of the topologies described above. This allows the size of each ordering domain composition to be reduced so that its performance is acceptable. Ordering domains can be connected in many ways (Figure 7). Because the inter-group routers impose no restriction on how groups can be connected, it is possible to construct very complex topologies to implement ordering domains efficiently.

One common way of structuring fault-tolerant distributed systems, particularly real-time systems, is as a pipeline of communicating process groups. A pipeline is typically composed of a number of pipeline “stages,” with each stage corresponding to a single ordering domain. Each stage in the pipeline receives data from preceding stages, processes it, and sends the results to succeeding stages (Figure 8). Of course, there are many other ways

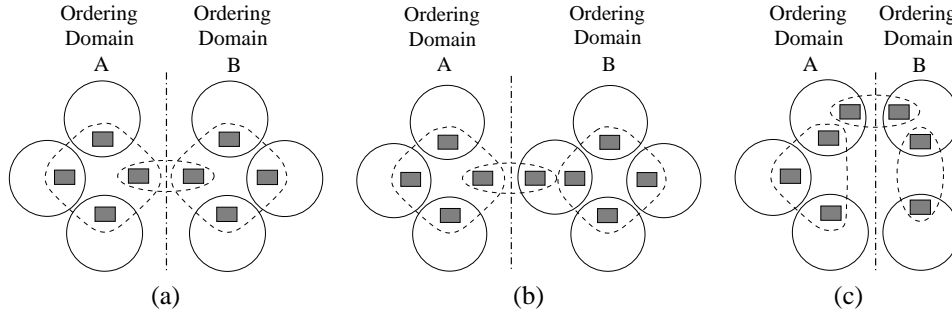


Figure 7. Methods of connecting ordering domains. Only the groups and inter-group routers are shown.

to compose ordering domains into complete systems, depending on the application’s communication requirements.

5. Performance Evaluation

In order to evaluate the scalability of the inter-group router architecture, we performed a trace-driven simulation of groups composed using inter-group routers for varying group sizes and 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.

5.1. Simulation Background

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 [1], a group multicast protocol we have developed which provides predictable, atomic, causal and totally ordered message delivery with real-time guarantees. The simulation modeled the entire protocol stack of each host, including the RTCAST, IP, and Ethernet protocols, the inter-group router protocol, and the complete physical network. The models included the actual processing time required for each network protocol, plus the propagation delay of the physical network. Based on our previous performance evaluations of the RTCAST implementation, we tuned the simulation to accurately reflect the real-world performance of running RTCAST on a Pentium-based testbed. In addition, in all simulations the message delivery order was verified to ensure that the end-to-end delivery semantics were correctly enforced. In all cases, messages were delivered in the correct order to all processes.

Since the performance of RTCAST compares very favorably with the published performance data of other group multicast protocols (most notably Totem [15] and Horus [18]), we expect that our simulation results will be applicable to systems built using protocols besides RTCAST. Figure 9 shows how the performance of a real RTCAST

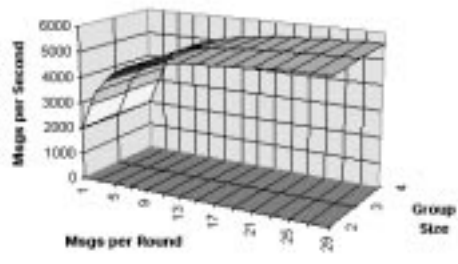
group implementation compares to an equivalent simulated RTCAST group. The simulation gives performance results which almost exactly match the actual implementation.

5.2. Simulation Results

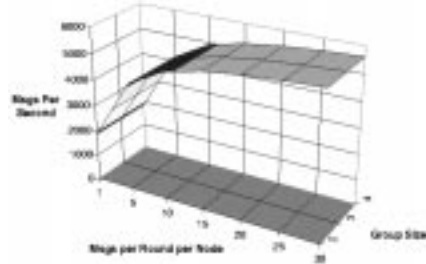
To compare the scalability of a single-group vs. a composed system, we simulated a single process group, a paired group composition, and a four-group multi-group composition. Each group was composed on a separate LAN, as was the inter-group communication protocol. For the inter-group protocols, we implemented a simple sender-based FIFO protocol for the paired group composition and used RTCAST for the multi-group composition.

In all tests, processes were distributed equally among all groups in the system. Messages were generated in a batch and sent by the application immediately after the logical token was released at that node, ensuring the worst-case latency for each message. Inter-group messages were either broadcast to all groups or sent to one remote group with equal probability.

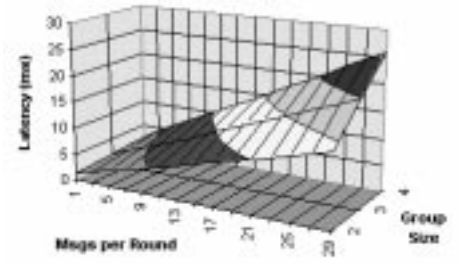
We focused on two areas of protocol performance in our simulations: message latency, and aggregate system throughput. Figures 10a and 10b show the average worst-case latency of messages in both a single group and the four group composition for two different message generation rates. Note that even when 100% of the messages in the four-group composition are sent to all processes in all four groups, achieving the same degree of communication as a single process group, the group composition still shows lower latency than the single process group. When fewer than 100% of the messages are sent to other groups, the performance of the system improves still further: with 20% inter-group messages, the latency of a 36 member four-group system is comparable to the latency of 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 may provide substantial performance benefits to large applications.



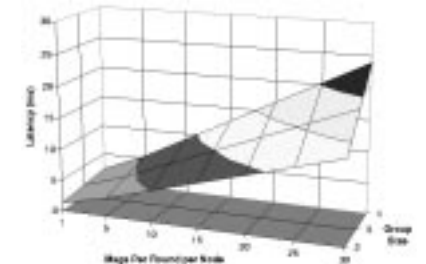
(a) implementation maximum message throughput



(b) simulation maximum message throughput



(c) implementation avg. message latency

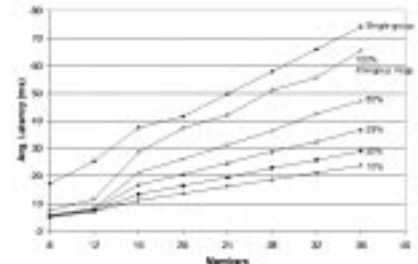


(d) simulation avg. message latency

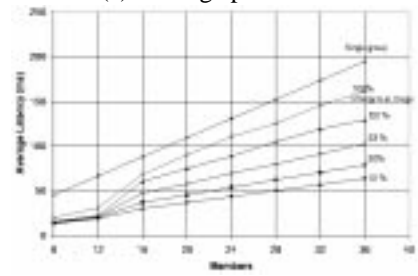
Figure 9. Comparison of RTCAST implementation and simulation, using 1-byte messages.

Although these results may seem counter-intuitive, there is a very simple explanation. In RTCAST, as in other protocols based on a logical token ring, message latency typically increases linearly with respect to the size of the group. At some point, the cost of delivering a message to a group of size n will exceed the cost of delivering a message to a group of size $\frac{n}{4}$, then to a group of size 4, and then to another group of size $\frac{n}{4}$. Although different types of group multicast protocols might not show the same immediate benefit from using multiple groups, at some point they would still likely reach a scalability barrier, beyond which more groups would be required to add additional processes to the system.

Figure 11 shows another way to view the data from these experiments. For the tests shown in this graph, 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. In this graph, we see that as more groups are added to the system, the aggregate throughput of the system increases almost proportionally, while the average latency for intra-group messages and some inter-group messages 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 which enforce end-to-end delivery semantics.



(a) 10 msgs per batch



(b) 30 msgs per batch

Figure 10. Worst-case message latency. Each line shows the results for a different proportion of inter-group messages to intra-group messages.

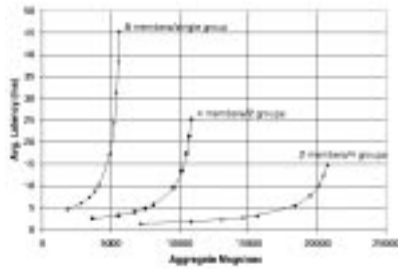


Figure 11. Aggregate throughput vs. worst-case latency for three basic topologies.

6. 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. Our approach is based on a novel architecture for modularly composing process groups together to form larger distributed systems. We also presented an analysis of end-to-end message delivery semantics in composed systems. Based on our analysis, a distributed system designer can determine what type of delivery semantics are required in each ordering domain of a composed system, and choose protocols which enforce the required delivery semantics efficiently. 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.

We are currently working on an implementation of the inter-group routers, and plan to test our group composition architecture in several applications to verify its scalability and performance. We also plan to perform further simulation studies of group compositions using other group communication protocols besides RTCAST. In the future we plan to add support for prioritized inter-group communication, and will study the performance of process groups composed with our architecture across wide-area networks such as the Internet.

References

- [1] T. Abdelzaher, A. Shaikh, F. Jahanian, and K. Shin. RT-CAST: Lightweight multicast for real-time process groups. In *Proceedings IEEE Real-Time Technology and Applications Symposium (RTAS '96)*, pages 250–259, June 1996.
- [2] 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.
- [3] Ö. Babaoğlu, A. Bartoli, and G. Dini. Enriched view synchrony: A programming paradigm for partitionable asyn-

- chronous distributed systems. *IEEE Transactions on Computers*, 46(6):642–658, June 1997.
- [4] Ö. Babaoğlu and A. Schiper. On group communication in large-scale distributed systems. *ACM SIGOPS Operating Systems Review*, 29(1):612–621, Jan. 1995. Also appears as Proceedings ACM SIGOPS European Workshop, September, 1994.
- [5] R. Baldoni, R. Friedman, and R. Renesse. Hierarchical daisy architecture for causal delivery. Technical report, Cornell University, 1996.
- [6] K. P. Birman. *Building Secure and Reliable Network Applications*, pages 308–309. Manning Publications Co., 1996.
- [7] J. Chang and N. F. Maxemchuk. Reliable broadcast protocols. *ACM Transactions on Computer Systems*, 2(3):251–273, Aug. 1984.
- [8] D. Dolev and D. Malki. The Transis approach to high availability cluster communication. *Communications of the ACM*, 39(4):64–70, Apr. 1996.
- [9] 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)*, pages 282–291, Seattle, WA, June 1997.
- [10] U. Fritzsche, Jr., P. Ingels, A. Mostefaoui, and M. Raynal. Fault-tolerant total order multicast to asynchronous groups. Technical report, Centre National de la Recherche Scientifique, Jan. 1998. To appear in SRDS '98.
- [11] C. Huitema. *Routing in the Internet*. Prentice Hall PTR, Englewood Cliffs, New Jersey, 1995.
- [12] S. Johnson, F. Jahanian, and J. Shah. Scalable group composition with end-to-end delivery semantics. Technical report, Dept. of Electrical Engineering and Computer Science, University of Michigan, 1998. Available from http://www.eecs.umich.edu/~scotttdj/papers/group_comp.ps.
- [13] H. Kopetz and G. Grünsteidl. TTP – a protocol for fault-tolerant real-time systems. *IEEE Computer*, 27(1):14–23, Jan. 1994.
- [14] S. Mishra, L. Peterson, and R. Schlichting. Consul: A communication substrate for fault-tolerant distributed programs. *Distributed Systems Engineering*, 1:87–103, 1993.
- [15] L. E. Moser, P. M. Melliar-Smith, D. A. Agarwal, R. K. Budhia, and C. A. Lingley-Papadopoulos. Totem: A fault-tolerant multicast group communication system. *Communications of the ACM*, 39(4):54–63, Apr. 1996.
- [16] S. Mullender, editor. *Distributed Systems*, chapter 5, pages 104–107. Addison-Wesley, second edition, 1993.
- [17] 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.
- [18] R. van Renesse, K. Birman, and S. Maffei. Horus: A flexible group communication system. *Communications of the ACM*, 39(4):76–83, Apr. 1996.
- [19] P. Verissimo and C. Almeida. Quasi-synchronism: A step away from the traditional fault-tolerant real-time system model. *IEEE TCOS Bulletin*, 7(4), Dec. 1995.
- [20] P. Verissimo, P. Bond, A. Hilborne, L. Rodrigues, and D. Seaton. The extra performance architecture (xpa). In D. Powell, editor, *Delta-4 - A Generic Architecture for Dependable Dist. Computing*. Springer-Verlag, 1991.