0. Chapter 4

1. Chapter 4

2. Chapter 4

3. Chapter 4

4. Chapter 4

Chapter IV

Characterizing and Evaluating Performance Tradeoffs in Causal Multicasting in ATM Networks

4.1 Introduction

Asynchronous Transfer Mode (ATM) networks are connection-oriented, cell-switched networks with very small packet (called cell) size designed for high-speed applications that require large amounts of bandwidth and low delay variance. ATM networks are becoming more popular and will greatly benefit several applications such as multimedia, transaction processing, and large file transfers.

Multicasting is important in teleconferences and other multimedia applications, as well as applications such as partially replicated databases. A message is *multicast* if it is sent to more than one recipient. In a multicast environment, messages should maintain their cause and effect relation (causal order), i.e., message delivery should preserve the cause and effect relationship. In addition, messages that are not related by cause and effect, i.e., concurrent messages, should be received in the same order by all recipients.

Multimedia messages also have a real-time constraint; they must be received by the destination site by a certain time after which their data cannot be used.

In a multimedia conference call, where each "speaker" is heard by multiple sites, for greater efficiency the messages should be multicast from the speaker to the other participants. The multicast messages need to have their cause and effect relations preserved. A causal order algorithm is required to preserve the causal order of the messages. Multicasting without causal ordering and real-time constraints has been studied in ATM networks [15,48] and causal order without multicasting has been studied [38,49].

Our objective is to establish boundaries to determine, given a multicast network topology, which causal ordering method is most efficient. To that end, we focus on two algorithms with different properties, such as overhead, and type of causal ordering provided. We present a framework to characterize and evaluate the overhead and performance of causal multicast algorithms and use this framework to evaluate the two multicast algorithms.

The rest of the chapter is organized as follows. Section 2 contains the system model, formal definitions, an overview of multicasting in ATM networks, and a description of previous work on causal multicasting. Section 3 presents a framework to characterize and evaluate the performance of multicast algorithms. Section 4 presents two

algorithms for causal multicasting whose performance will be analyzed and compared. Section 5 summarizes this chapter.

4.2 Background

4.2.1 System Model

The system is comprised of a set of nodes that are linked to each other through a communication network. ATM switches comprise the nodes and the links or edges are the links between the ATM switches. The system is modeled by a directed graph, $G = (V, E)$, where V is the set of vertices in the graph representing nodes in the system (i.e., ATM switches) and E is the set of edges between vertices representing communication links. The communication network supports multicasting from a source node to a set of destination nodes and determines the route for the multicast messages. A route for a multicast forms a subgraph of G denoted as G_{mcast} and referred to as the multicast topology, which is typically a tree with the source node as a root. All leaf nodes are destination nodes, however, destinations for the multicast can be non-terminal nodes, as well. N is used to denote the number of recipients for a multicast.

A *multicast* message is a message that has more than one recipient, while a *unicast* message has only one recipient. *Broadcast* messages are a special case of multicasts in which message recipients are all sites in a domain.

Recall the definition of causal order and delta causal order from Chapter 3. A *causal multicasting* algorithm is an algorithm which enforces causal ordering among multicast messages. This is done by adding control information about causal dependencies to every message and maintaining state information at each site. The causal order is enforced by delaying the delivery to the application of messages that arrive out of order, usually until the missing messages have arrived.

4.2.2 Multicasting in ATM Networks

Asynchronous Transfer Mode (ATM) networks are connection-oriented, cell-switched networks, which are fundamentally different than traditional store-and-forward networks [37]. The ATM protocol is divided into three layers, the physical, the ATM, and the ATM adaptation layer (AAL). The physical layer is similar to the physical layer of the OSI model. The ATM layer is responsible for transporting the cells, and the AAL is responsible for supporting different types of traffic with different characteristics. Layers above these three represent higher level protocols, such as application layers [9].

There are two issues involved in multicasting on ATM networks: cell replication and routing. At the lowest level, in order to multicast, one cell must be delivered to multiple output ports. The cell must be replicated within the switch, in order to deliver the cell to different output ports on the switch. This typically is dealt with by the switch hardware and firmware [15,48].

The other issue is where to send, or route, the cells. At the switch level, cells must be routed to the proper output ports within a switch once they have been replicated. At the network level, the cells must be routed from one switch to another in order to efficiently reach their destinations. The problem of finding an optimal path or route given a network description has been studied in [24,35,40]. Often there are multiple paths and routes that can be used; so a goal is to minimize the number of paths or number of cells that must be transmitted. Much of the research on multicasting in ATM networks is focused on the issues of replication and routing.

4.2.3 ATM Networks and Causal Multicasting

Multicasting on ATM networks is important, since such networks are attractive for multimedia applications. Multicasting is important in multimedia applications because applications, such as conference calls, take advantage of the ability to send the same data stream to multiple destinations. Such multimedia applications require a large bandwidth. ATM networks are capable of offering a high bandwidth, as well as providing guaranteed quality of service. There are several issues that need to be addressed in order to utilize a causal multicast algorithm in an ATM network. One of the most prominent features of ATM is the small cell size, 48 bytes for the payload (data) and 5 for control information (53 bytes total). Causal multicast algorithms typically impose a per message overhead because they append control information to each message, The delta causal multicast method, the algorithm in this chapter with the least overhead, imposes an overhead of 2

vectors of *N* elements each with every message, for *N* recipients. Each element contains a timestamp, which requires a long integer, four bytes, to store. In an ATM network, if the multicast algorithm is implemented at the ATM layer, by using all of the payload space in an ATM cell, at most only 6 sites could be included in a multicast group (2 vectors * 6 timestamps/vector * 4 bytes/timestamp). Obviously, this is too restrictive, since only 6 sites at most could be supported by consuming all the space for control information leaving no space remaining in the payload for the actual multicast data.

Therefore, the causal multicast algorithm must be implemented at a level higher than the ATM layer. The ATM adaptation layer (AAL) contains two sublayers, the segmentation and reassembly sublayer (SAR) and the convergence sublayer (CS). The SAR sublayer is responsible for dividing large packets from the higher layers into ATM cells at the transmitting end, and the reverse at the receiving end. The CS sublayer performs service dependent functions, such as message identification and time/clock recovery [9]. Causal multicast algorithms would be appropriate for the CS sublayer or higher. Causal ordering would be based on the application layer message, independent of the cell size. It is more appropriate to perform causal ordering at the message or packet level, as opposed to the cell level. Performing causal ordering at a higher level is more cost efficient since the ratio of data to control information is greater. Thus there is a higher space utilization and less waste.

4.2.4 Previous Work

Causal ordering algorithms impose both computational as well as message overheads. The computational overhead comes from the underlying algorithm that enforces causal order and varies from algorithm to algorithm. The message overhead comes from the extra control information that must be transmitted with each message. This information can be in the form of time vectors [38,49] or message histories [7]. ISIS, the first system to support causal multicasting, sends complete message histories with each message, which may cause a significant overhead. It enforces causal order of messages and additionally can enforce a total order on all messages, including concurrent messages. It does not support real-time constraints.

Mostefaoui and Raynal [33] provide a low cost algorithm for causal multicast; however, it assumes that all messages are broadcast to all sites. This allows every site to have current information about the causal relations between messages. In addition, the algorithm requires periodic resynchronization messages to be sent. If the communication medium is a bus based medium, like an Ethernet, only single broadcast is needed for the resynchronization message. This algorithm does not support any real-time constraints.

The Psync protocol by Peterson, Buchholz, and Schlichting [36] preserves the partial order between messages, and supports fault-tolerance. It does not impose a total order between messages at different sites, nor does it support real-time constraints.

The causal ordering algorithm of Schiper, Eggli, and Sandoz [38] insures that causal order relation between message is maintained. However, this method has a significant overhead as O(*N*) vectors must be transmitted and it does not support real-time constraints. This method captures the casual order relations between messages by maintaining a dependency matrix which consists of a dependency vector for each site. An element *j* of a vector *i* is a logical timestamp for a site and represents the last known state of site *j* for site *i*, or is state of site *j* that site *i* depends on. While complete transitive causal order relations can be traced, real-time constraints are not supported.

The causal ordering algorithm of Adelstein and Singhal [1] has a much lower overhead and maintains "delta causal order" [49], in which the causal order relations only exist within a limited delta time interval. It only prevents causal order violations due to the "triangle inequality," in which the causal dependency chain has a maximum length of two. However, rather than supporting multicasting directly, it simulates sending a multicast message to *N* sites with *N* separate unicast messages.

4.3 Characterizing Multicast Cost

In this section we characterize and evaluate the cost of multicasting for various causal order algorithms. We identify the underlying tradeoffs with respect to the cost and performance for these algorithms. We study these algorithms for various topologies since the underlying multicast topology has a direct impact on the cost of transmitting data across the network. A naive approach to computing this cost is to multiply the

message size, including the overhead from the causal ordering scheme, by the number of recipients of the message. However, this only reflects the cost of the transmission to the recipients, and does not reflect the total cost associated with propagating a message through a communication network. This implies all multicasts with the same number of recipients have identical cost. Yet two multicasts with the same number of recipients can impose very different costs on the network in terms of volume of information. Figure 37 shows two multicasts that send a message to the same number of destinations, but use different routing. In multicast A, approximately 90% of the path is common to all the destinations and is shared, while only 10% is shared for multicast B.

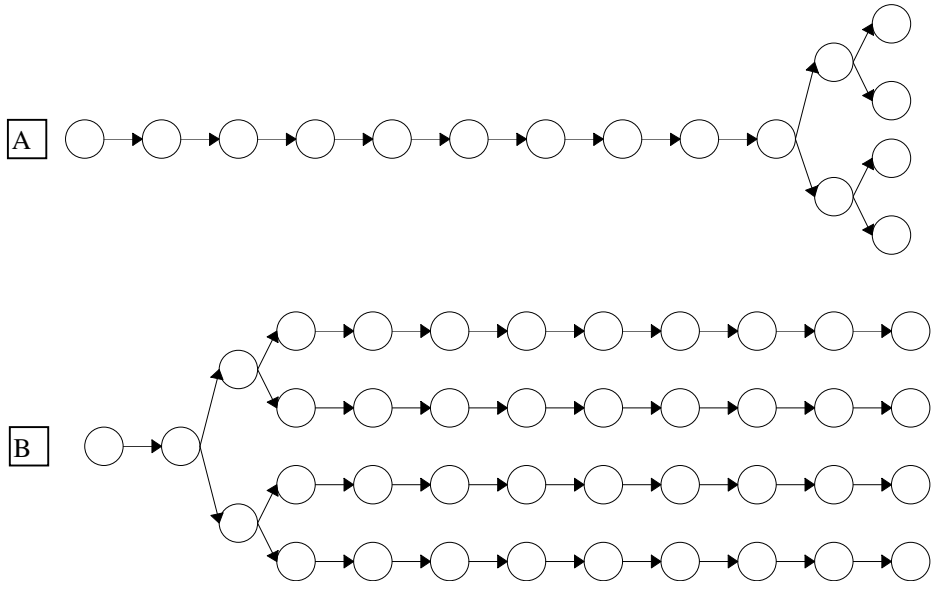

Figure 37: Two multicasts with different routing topologies

Intuitively, multicast A is more efficient than multicast B. Since there is less duplication, fewer sites are required to (re)transmit data. However, it is possible that an algorithm that uses multicast A requires a much higher bandwidth than one that user multicast B. In which case, the entire path in multicast A is consumed by the one source, tying up the network, while in multicast B, more of the network is used but because the bandwidth is lower, the paths can be shared by other applications. To characterize the relative efficiency of a multicast, we need to calculate the cost that a causal order algorithm imposes upon the network. The cost must reflect the total volume of information that is sent through the network and this is related to the underlying topology. The cost value can be used to help minimize the load imposed by a source site upon the network. By reducing the overall load, the network can support a greater number of sites transmitting.

Our framework to compute this cost consists of the following. First, we count the number of message hops taken by a multicast algorithm for a given multicast topology. Then we define the efficiency of a multicast topology as the relative saving of number of hops. This is one minus the number of message hops of a multicast divided by the number of hops of a unicast. We then compute the total cost of the algorithm as the total volume of information transmitted to factor in the control information.

4.3.1 Total Network Hops Used (S)

The cost of a multicast depends on the underlying multicast topology of the communication network and the causal multicast algorithm used. In order to

characterize the cost, a variable that characterizes the underlying topology and algorithm is needed. To that end, we will determine how many messages are sent in a multicast, which allows us to calculate the volume of data that is transmitted through the network for each multicast message.

L represents all the links in the multicast tree, l is a link in the multicast tree, and cnt_l is the count of how many times messages are sent over link l. S can be expressed as follows:

$$S = \sum_{\forall l \in L} l * cnt_l \qquad [6]$$

Total network hops is defined as the total number of messages sent through a network in order to transmit a multicast message. If a multicast algorithm sends only one message then S is the number of links in the multicast topology. If the multicast algorithm sends more than one message, then S is the number of times each link is used. S is sensitive to the connection topology.

A multicast message can be transmitted to all its recipients simultaneously (i.e., a multicast), to each one sequentially, or by some combination of both. The computation of S depends greatly on how the message is transmitted. The following two paragraphs describe how to compute S for two different multicast algorithms for the cases of one simultaneous transmission and many sequential transmissions, using the network topologies shown in Figure 37.

A message is transmitted to all recipients simultaneously in a single multicast. When the message is transmitted, it splits whenever branching occurs in the multicast tree. For example, in Figure 37, a split occurs at the ninth and tenth nodes in multicast A and the second and third nodes in multicast B. The *S* value for multicasting, denoted by S_M, is computed by summing the total number of links in the multicast tree. Since only one message passes through each link, any path that is shared only counts once towards the value of S_M. Then,

$$V = (v_1, v_2, \ldots, v_N)$$

$$E \subseteq V \times V$$

$G = (V, E)$ is a graph that represents the communication network. $G_{top} = (V_{top}, E_{top})$ is a subgraph of G and is a tree with node v_1 as its root. G_{top} represents the multicast topology for the communication network. Given G_{top}, S_M is defined as,

$$S_M = \left| E_{top} \right|. \qquad [7]$$

For multicast A, S_M is 15, and for multicast B, it is 39. S_M denotes how efficient the routing is -- as the routing becomes more efficient, more paths are shared and the value of S_M decreases. An upper bound for S_M is S_U, described next, which occurs when a multicast is implemented as a series of unicasts.

In the second method, a multicast message is transmitted as a series of unicast messages, one for each destination. The *S* value for the set of unicasts, denoted as S_U, is computed

by summing the number of links the message passes through for each sender to recipient pair. Given a network multicast topology with one source, v_1, and multiple destinations: $Dest = (d_1, d_2, ..., d_m)$, where $m < n$ and $Dest \subseteq V_{top}$, and *Dest* represents recipients of the multicasts.

$Path(j)$ = length of path from v_1 to j. Then,

$$S_U = \sum_{\forall j \in Dest} Path(j). \qquad [8]$$

The value of S_U for both topology A and topology B is 44, since both topologies have 4 recipients, each with a path length of 11. S_U is independent of how much path is shared among destinations in a multicast topology.

4.3.2 Efficiency of a Multicast (E)

The total cost of sending a multicast is affected both by the overhead of the control information sent by the causal order algorithm and the efficiency of the underlying multicast topology. Intuitively, a multicast topology is more efficient if fewer messages are transmitted over the links to perform a multicast over the communication network. As the efficiency increases, more links are shared by the destinations and fewer messages are transmitted over every link in the network. This results in a lower total volume of data that is transmitted through the network. The cost of sending a multicast message through the multicast topology depends upon the efficiency of the multicast topology.

The efficiency of the multicast is expressed with respect to the cost of sending the same message to the same destinations using only unicast messages.

The efficiency of a given multicast topology is defined as follows:

$$Efficiency = 1 - (\frac{S_M}{S_U}). \qquad [9]$$

For example, in Figure 37, for multicast A, S_U = 44 and S_M = 15, so the efficiency is 66%, and for multicast B, S_U = 44 and S_M = 39, so the efficiency is only 11%. Note that in multicast A, 90% of the path is shared, while in multicast B only 10% of the path is shared. Thus, there is a direct relation between the efficiency and the amount of shared path in a multicast topology. Causal ordering algorithms impose a per-message overhead in terms of the control information sent. In order to evaluate the cost of causal order algorithms, it is necessary to know the cost of sending a message through the network and the efficiency of the particular network topology. We now compute the transmission cost in multicasting a message.

4.3.3 Transmission Cost (C)

The transmission cost of sending a multicast message through a network includes the total volume of information that is sent through the network in multicasting a message. The cost represents how much work is done by the network. As a network transmits a greater volume of information, issues like congestion and flow control, as well as

bandwidth requirements and allocation become more important. The transmission cost takes into account each link in the network that is used to transmit the message. In performing a multicast transmission, the total network cost is:

$$cost = S * (msg_size) \;\; where \;\; msg_size = data + overhead . \qquad [10]$$

The cost is in terms of total bits transmitted over all links in the multicast tree, since *S* represents the total number of links a message must pass through and the *msg_size* represents the amount of data that must be passed through a node, including the data and the overhead from the underlying causal ordering algorithm.

4.4 Cost Analysis of Two Causal Multicast Algorithms

In this section we perform cost analyses of two algorithms to determine the cost-performance tradeoffs. The first algorithm, the delta causal multicast method [1], provides a restricted form of causal ordering, known as the *triangle inequality*, supports real-time constraints (delta-causality), has a low overhead, and enforces identical ordering of concurrent messages for all sites. However, it performs a multicast by a series of unicasts. The second algorithm, the causal order multicast method [38], maintains all causal order relations. However, it has a large overhead, it does not support real-time constraints, and does not order concurrent message.

4.4.1 Delta-Causal Multicast Method

The delta-causal multicast method is a low-overhead method that provides a restricted form of causal ordering called "delta-causal ordering" [49]. This method handles violations in causal ordering due to the *triangle inequality* [1]. The triangle inequality occurs when a message that passes through an intermediate site takes less time to arrive at its destination than one directly sent to the same destination. Most violations of causal ordering in a real-time environment are likely to be due to the triangle inequality, because a message sent directly to its destination that takes longer than one that traverses multiple hops is likely to have exceeded its deadline.

Each site maintains a dependency matrix in which an entry at *(i, j)* represents the timestamp of the last known timestamp of site *j* as far as site *i* knows. The timestamp is based on a physical clock that is kept synchronized with the other sites within some error tolerance by a clock synchronization algorithm [19, 31]. For each message sent from site *i* to site *j*, two vectors are appended to each message. The vectors are row *i* and column *j* of the dependency matrix. Upon arrival of the message, the vectors are compared to the corresponding ones at the destination in order to detect missing messages that the current message is causally dependent on. Concurrent messages can be identically ordered based on the clock value of their timestamps. The algorithm implements multicast support by having the underlying communication system implement multiple unicasts. Since two vectors of length *N* are added to each message, the

overhead for each message is $2*N$. Therefore, the cost of this algorithm, denoted as $cost_U$, is:

$$cost_U = S_U*(data_size) + 2NS_U. \quad [11]$$

4.4.2 Causal Order Multicast Method

The causal order multicast method [38] provides complete causal ordering of messages between all sites; however, the message overhead of this method is much greater than that of the previous one. A dependency matrix, like the one used in the previous method, is maintained, and the entire matrix is transmitted with each message. The matrix is merged with the recipient site's matrix and missing messages are detected by comparing the values in the matrix in the message with the values of the matrix at the recipient. Messages that depend on other messages that have not yet arrived are held until those missing messages have arrived. Concurrent messages are not guaranteed to be delivered in the same order at every site by this algorithm. Since the entire dependency matrix must be sent with each message, which consists of N vectors of length N, the overhead for each message is N^2. Therefore, the cost of this algorithm, denoted as $cost_M$, is:

$$cost_M = S_M*(data_size) + S_MN^2. \quad [12]$$

4.4.3 Comparison of the two methods

Comparing the costs of equations [11] and [12], it is clear the delta causal multicast method is more expensive than the causal order multicast method when:

$$S_U\,(2N + data_size) > S_M\,(N^2 + data_size). \qquad [13]$$

By using the definition of efficiency, equation [13] can be rewritten as:

$$E > 1 - \frac{2N + data_size}{N^2 + data_size}. \qquad [14]$$

Regrouping equation [14] yields:

$$N^2 + N^2E - 2N - E*data_size < 0. \qquad [15]$$

When both methods have equal cost, equation [15] can be expressed as:

$$N^2 + N^2E - 2N - E*data_size = 0. \qquad [16]$$

Applying the quadratic formula to equation [16] yields:

$$N = \frac{2 \pm \sqrt{2^2 + 4(1+E)(E * data_size)}}{2(1+E)} \qquad [17]$$

Thus, given E and the *data_size*, N can be determined, which represents the number of recipients where both methods are of equal cost. Values of N greater than the value computed from equation [17] represent network configurations in which the delta causal multicast method has a lower cost. Similarly, the causal multicast method has a lower

cost when values of N are less than the value computed from equation [17]. Note that the quadratic formula produces two roots to the equation; however, negative values for N have no meaning and will be ignored.

As a simplifying assumption, let us first assume that *data_size* is equal to zero. This allows us to only focus on the second term of the costs in equations [11] and [12], which represent the cost of the overhead of the algorithms. It is clear that the causal order multicast method is more efficient than the delta causal multicast method if $S_U > S_M N/2$, which can be expressed as $2/N > S_M/S_U$. Recall that efficiency is defined as $1-S_M/S_U$. Therefore, when *data_size* is equal to zero, if $2/N > 1-E$, then the causal order multicast method has less overhead (sending the full dependency matrix in one multicast is more efficient than sending 2 vectors for N unicasts); otherwise, the delta-causal ordering algorithm has less overhead. Assuming *data_size* is zero allows a low cost analysis to be performed and provides a lower bound on the N versus E curves. Any point below that curve represents a value of N and E in which the causal multicasts method has the least overhead. In addition, if the amount of data is small in comparison to the control information, this equation can provide an approximation of the cost.

Given E and a non-zero *data_size*, equation [17] gives a value of N where the two methods are of equal cost. Figure 38 plots the break-even value of N for values of E from 0 to 1, for various values of *data_size*. For the non-zero *data_size*, the value of *data_size* was chosen to be 100 and 1000, in order to model small and large messages. The zero *data_size* curve is also plotted to show a lower bound. Points above the curve

represent values where the delta causal multicast method is cheaper, while points below the curve represent values where the causal order multicast method is cheaper. Note that the causal order multicast method tends to have a lower overhead than the delta-causal multicast method when there is a lot of path sharing which implies higher efficiency.

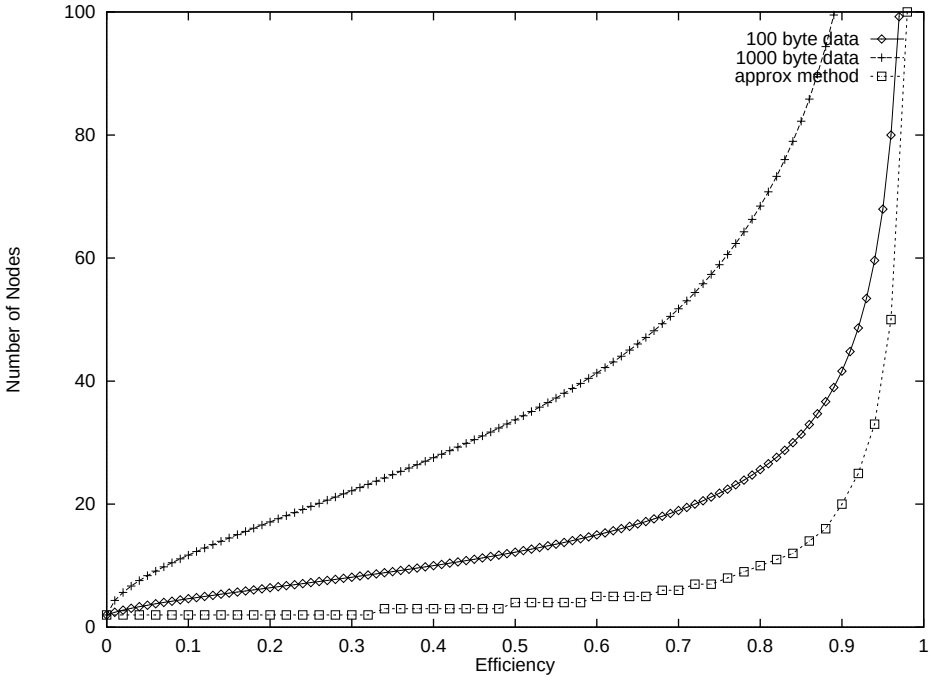

Figure 38: Break-even curve for N vs. E.

Comparing the two methods, it is evident that assuming a zero *data_size* captures the basic shape of the curve and gives a lower bound to the *E* vs. *N* curves given by equation [17]. As *data_size* decreases, the curve goes down, converging when *data_size* equals zero. Anything below the zero *data_size* curve represents a point where the causal

multicast method is more efficient than the delta causal multicast method and is a tight bound. As *data_size* increases, the difference between the zero *data_size* curve and other curves increases. To determine the least expensive method given a network topology and *data_size*, *E* and *N* can be determined and compared to the corresponding point on the zero *data_size* curve. The zero *data_size* curve is easy to compute and serves as a lower bound, indicating when the causal order multicast method is the least expensive. If the point corresponding to the calculated values of *E* and *N* is not below the zero *data_size* curve, then the point can be compared to the curve computed by equation [17] for given value of the *data_size* to determine which algorithm to use.

We also observe that for small numbers of recipient nodes, in general the delta causal multicast method is cheaper. As the efficiency of the topology increases, the causal order multicast method becomes cheaper than the delta causal multicast method. The curve rises exponentially as the value of efficiency approaches one. In addition, as the size of the data increase, the curve tends to be higher, meaning that more recipient nodes can be supported by the causal order multicast method at the same efficiency level.

4.5 Summary

In this chapter we developed a method of expressing the efficiency of a multicast for a given network topology. The efficiency describes the amount of sharing among the network paths. The greater the number of shared paths, the higher the efficiency. Given

the efficiency and the data size, we developed a framework to characterize and evaluate the cost of causal multicast algorithms.

We discussed two methods to support causal multicasting on ATM networks, namely, the causal order multicast, delta causal multicast methods. The causal order multicast method supports causal ordering but does not support real-time constraints and has a high overhead, $O(N^2)$. The delta causal multicast method eliminates violations in causal order only due to the triangle inequality; however, it supports a time restricted form of causal ordering known as delta-causal order and has a much lower overhead than the causal order multicast method, $O(2N)$.

Utilizing the developed framework for evaluating the causal multicast algorithms, we compared the costs of the two methods to determine the conditions under which each had the best performance and derived an equation that can be used to determine which method has the least cost for a given set of parameters. By using the least cost algorithm for a given configuration, network bandwidth usage is reduced, which allows more sites to use the network

5. CHAPTER 5

Chapter V

Partition Method for Real-Time Causal Multicast

5.1 Introduction

The delta causal multicast method discussed in the previous chapter supports multicasting by performing a series of unicasts. If the underlying communication network has multicast capability, it is more efficient to transmit a messages to multiple destinations as a single multicast message as opposed to multiple unicast message. For a multimedia conference call application, as a participant "speaks," the data is transmitted to the other participants in the conference. This type of transmission greatly benefits if the underlying network supports multicast capability. The problem of providing message delivery in causal order still exists in a multicasting environment. Therefore, a causal multicast algorithm will have a higher efficiency for message transmission in a multicast environment than a similar algorithm in a unicast environment.

Chapter IV presented a framework to evaluate the efficiency of causal multicast algorithms and used it to compare the efficiency of two algorithms. In this chapter, we propose a new algorithm, called the partition method, that ensures the delivery of

messages in causal order while maintaining real-time constraints, has a low overhead, and supports multicasting. The partition method, proposed in this chapter, supports multicasting with low overhead. It breaks up the multicast into groups of smaller multicasts called partitions, in which sites in the same partition are close together with respect to the network. Each partition receives its own multicast, which contains control information specific to its destinations, similar to the delta causal multicast method. It is designed to have good performance for topologies that can be divided into small clusters or partitions of nodes with mostly common paths.

We use the framework established in the previous chapter to analyze the efficiency of this method and compare the cost of the partition method with the causal order multicast method and the delta causal order multicast method presented in the previous chapter. The costs are computed for several sample topologies that represent different efficiencies.

The rest of the chapter is organized as follows. Section 2 presents the partition method. Section 3 performs a cost analysis and compares the cost of the partition method with two other causal multicast methods for several sample topologies. In addition, this section analyzes the impact of the number of elements in a partition on the cost.

5.2 Partition Method

The partition method is a low cost causal order algorithm that directly supports multicasting. It maintains real-time delta causality constraints and has a lower cost than sending the entire dependency matrix with each message.

5.2.1 Basic Idea

Figure 39 shows an example of a network topology in which it is easy to partition into two multicasts. The first partition includes sites B and C, and the second partition includes sites D and E.

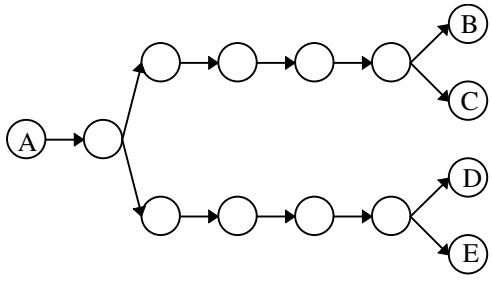

Figure 39: Easily partitioned network topology

The partition method divides the multicast topology into groups or partitions (refer to Section 5.2.5 for a description of how partitions are formed). The multicast is broken into as many multicasts as there are groups; the sender sends a separate multicast for each group. Every site within a partition receives the same multicast message, including the control information. Sites in different partitions receive different multicast messages.

The data is the same for all sites, but the control information differs from partition to partition. The result is that the original multicast message is replaced by several smaller multicasts.

Control information is appended to every message. The control information in a message is similar to that in the delta causal multicast method; that is, two vectors are sent with every message from site *i* to site *j*, specifically row *i* and column *j* of the dependency matrix at site *i*. Since the second vector, i.e., column *j*, is specific to the destination, multicasting a message to all of the recipients in a partition requires adding an additional vector for each site. So for the partition method, for each partition, row *i* of the dependency matrix is sent, as well as every column *j*, for a node *j* in the partition receiving the multicast. When a site *j* receives a multicast message from site *i*, it extracts two vectors, row *i* and column *j*, from the control information that was appended to the message.

The delta causal multicast method is a special case of the partition method in which each partition contains only one element. The worst case occurs when all nodes are contained in one partition, which is similar to the delta causal multicast. For *N* recipients, all in one partition, $N + 1$ vectors are sent, with each vector containing *N* element, which is an overhead of $N^2 + N$. In this case, the overhead cost is slightly higher than that of the causal order multicast method which sends the full dependency matrix with each message

and has an overhead of N^2. However, if partition sizes and members are properly selected, the cost is less than either of the previous approaches.

5.2.2 Data Structures and Notations

Each site maintains the following variables:

- Dep_i represents an *NxN* dependency matrix at site *i* in which each element *(x, y)* of the matrix represents the timestamp of the last message site *x* sent to site *y*, as far as site *i* knows.

- *Current_time* represents the current time indicated by the local real-time clock at a site. Every site has a clock and constantly updates it by an underlying clock synchronization algorithm.

- *Min_wait* is a variable that denotes the amount of time necessary to hold a message to compensate for the network transit time.

- Δ represents the time the data in a message is useful and is defined to be $< 3 *$ *min_wait*. For example, an upper bound for Δ, for audio data, is 100 milliseconds.

- *Message-queue* is a priority queue to hold messages that have arrived from the communication medium but have not yet been delivered to the application layer. The queue is ordered in increasing order of the message timestamp.

- *Msg* is the multicast message; it consists of the data or payload, denoted as *msg.data*, and up to *N* vectors added by the causal order algorithm, denoted as $msg.vect_x$, where $1 \le x \le N$.

5.2.3 The Algorithm

Multicasting a Message from Site i to Recipients in ϕ[1]

```
SendMessage(msg, i, φ){
        ∀j∈φ, Dep_i[i,j] := current_time
        ∀ x (msg.vect_1[x] := Dep_i[i,x])          /* row i of dependency matrix */
        n := 2
        FOREACH z in (φ)
                ∀ x (msg.vect_n[x] := Dep_i[x,z])  /* column j of dependency matrix */
                n := n + 1
        ENDFOREACH
        Multicast message to sites in φ.
} /* end function */
```

Receiving A Message From Site i At Site j

```
ReceiveMessage(msg, j) {
        IF (msg.vect_1[j] + Δ > current_time) THEN
                Discard message.                   /* the msg is past its deadline */
                RETURN
        ENDIF
```

---

[1] where ϕ is the set of destinations for the multicast

```
        /* insure the message is not delivered until      */
        /* at least min_wait time units after being sent */
        wait (MAX(0, (msg.vect1[j] + min_wait - current_time)))          /* Rule 1 */
        Extract vector corresponding to column j as msg.vect2.
        CheckAndDeliver(msg, j)
} /* end function */
```

Each recipient site is concerned only with two vectors. The first corresponds to row *i* of the sender *i*'s dependency matrix. The second corresponds to column *j* (for a particular recipient site *j*) of sender *i*'s dependency matrix. Since multiple vectors are appended to a message, sites must be able to identify which vector corresponds to which column of the dependency matrix. This can be done by including an ID field with each vector or by agreeing upon some *a priori* order for the vectors when the multicast tree is formed.

```
CheckAndDeliver(msg, j) {
        /* compare the message with the Dep matrix of the destination site (j) */
        Put message msg in queue.
        IF (DependencyExists(msg, Depj)) THEN                    /* Rule 2 */
                Return
        ENDIF
        DeliverMessage(msg)              /* deliver message with no dependencies */
        /* check other messages in queue and deliver any that do not have dependencies
*/
        /* since other messages may have been waiting for this message. */
        FOREACH x in (message-queue)
                IF (NOT (DependencyExists(x, Depj))  THEN
```

```
                    DeliverMessage(x)
                ENDIF
            ENDFOREACH
} /* end function */


DeliverMessage(msg) {
        /* deliver message and merge sending site's dependency vector into local matrix
*/
        Remove message msg from queue.
        FOR x := 1 to N DO
                IF (msg.vect1[x] > Depj[i,x]) THEN
                        Depj[i,x] := msg.vect1[x]
                ENDIF
        ENDFOR
} /* end function */


Boolean DependencyExists(msg, matrix) {
        /* a dependency is represented by a timestamp in the message that is later than */
        /* the corresponding one in the dependency matrix, provided that that */
        /* timestamp is no older than by Δ (i.e., missing message has not expired) */
        FOR x := 1 to N DO
                IF ((msg.vect2[x] > matrix[x,j]) AND (current_time - msg.vect2[x]) < Δ)
                THEN
                        RETURN true
                ENDIF
        ENDFOR
```

```
        RETURN false
} /* end function */
```

Additionally, when a message is put in the queue, an asynchronous timer is set at MAX(msg.$vect_2$[x] + Δ) for all x. When the timer expires, the system checks if the message has any dependencies by calling the function DependencyExists(). The message is discarded if so and delivered otherwise. All other messages in the queue are checked for dependencies because undelivered messages may now become deliverable, if the queued message only depends upon the discarded or delivered message. Once the message is delivered (or discarded), the other messages on its dependency chain can be delivered.

The proof of correctness for the delta causal multicast method presented in Section 3.3.5 applies to the partition method. Since the delta causal method is a special case of the partition method in which all recipients are in the same partition, the proof can be generalized to apply to the partition method.

5.2.4 Correctness Argument

The proof of correctness for the partition method follows from the proof presented in Section 3.3.5. In this section, it was shown that 2HOP order can be maintained in a unicast environment by adding 2 vectors of length *N* to each message corresponding to row *i* and column *j* of the dependency matrix at site *i* for destination site *j*. In the

partition method, the function send_message() is used to send a multicast from site i to the sites in the set Φ, where $|\Phi|$ is k, and $k + 1$ vectors are added to the message. The first vector corresponds to row i of the dependency matrix of site i, and the remaining k vectors correspond to column j for all $j \in \Phi$. Each site j that receives the multicast message extracts the two vectors that correspond to row i and column j of the dependency matrix from site i and uses the vectors in the same way to enforce causal order as the algorithm presented in Chapter III. From Lemma 1 from Chapter III, it follows that each recipient can be shown to ensure 2HOP order. If 2HOP order is maintained between each individual sender-recipient pair, 2HOP order is maintained for all the recipients of a multicast, since a multicast can be viewed as a series of unicasts.

5.2.5 Selecting Elements in a Partition

Selecting nodes in a partition has a strong influence on the cost of multicasting the data in the partition method. In addition, the number of nodes in a partition also influences the cost. The efficiency of a partition is equivalent to the definition of efficiency from equation [9] except that S_U and S_M are only computed for the nodes in $part_i$. The following simple heuristic servers as a criterion to decide whether a node should be included in a partition. Let α be the minimum allowed efficiency for any partition. A node v_x can be added to partition i if the efficiency of the partition is greater than or equal to α.

The size of a partition plays an important roll in influencing the total cost of a multicast for the partition method. In Section 5.3.2 we analyze the effect the size of a partition has on the cost.

5.3 Performance Results

In this section, we compute the total network transmission cost using the causal order multicast, delta causal multicast and partition methods on four different topologies. For the partition method, three partitions of two recipients each were used (in Section 5.3.2 we example how partition size affects the cost). For the purpose of comparative analysis, we focus on some extreme cases with respect to efficiency, similar to the examples in Figure 37. In the multicast topology shown in Figure 40 the split occurs late and most of the path from the sender to the recipients is shared. In contrast, Figure 41 shows a multicast topology where the elements split as early as possible and there is no path sharing. The late split topology is well suited for multicasting, while the early split topology is not. Figure 42 shows a topology where the nodes quickly split into smaller subgroups and the subgroups split into individual recipients at the final stage. Figure 43 shows a topology in which there are two groups of recipient nodes, one group that is relatively "close" to the sender and another that is relatively "far" from the sender. This topology represents a less extreme case that may be more likely to occur in practice than the previous examples.

Since we are concerned with the overhead imposed by the algorithms, *data_size* is initially assumed to be zero in the examples. The results for non-zero *data_size* are presented in Table 2 and Table 3.

Cost Analysis

We now analyze the cost overhead of the partition method. Assume there are k partitions, where each partition contains q nodes (initially we assume there are an equal number of nodes in each partition). So the number of recipients of the multicast, N, is $q * k = N$. Since a separate multicast is done for each partition and since each partition can have different characteristics, including efficiency, one S value is not sufficient to describe the cost of the entire multicast. Therefore, unlike the previous two methods, a separate S value must be computed for each partition. The variable S_{Mi} is used to denote the S_M value for partition i. Therefore, S_{Mi} reflects the number of links that are required to send a multicast message to a partition i. The cost of sending a multicast message to the sites in partition i is obtained by multiplying S_{Mi} by the amount of data. The cost for each partition must be summed in order to compute the cost of the entire multicast. Using the definitions from Section 4.3.1, we define:

$$Part_i \subseteq Dest \text{, where}$$

$$\bigcup_i Part_i = Dest \text{, and}$$

$$Part_i \cap Part_j = \varnothing, \forall i, j, i \neq j .$$

$G_{top} = (V_{top}, E_{top})$ is a subgraph of G, and is a tree rooted at v_1, and represents the multicast topology. $G_{Pi} = (V_{Pi}, E_{Pi})$ is a subgraph of G_{top}, and represents the multicast topology for a partition i, where $E_{Pi} \subseteq V_{Pi} \; x \; V_{Pi}$, $V_{Pi} \subseteq V_{top}$, and $V_{Pi} = \{v_x, \forall x \mid \exists v_{dest} \in Part_i$ such that v_x is an element along the path from v_1 to v_{dest}, inclusive$\}$. S_{Mi} is defined as follows,

$$S_{Mi} = \left| E_{Pi} \right| . \qquad [18]$$

Assuming each partition contains the same number of sites, say q, each message contains $q + 1$ vectors with N elements per vector (if partition sizes are not of the same size, a different value of q can be used for each partition). So the control information overhead for a partition is described as,

$$overhead = (q + 1)\, N = Nq + N. \qquad [19]$$

Therefore, the multicast cost for a partition is:

$$(Nq + N + data_size)\, S_{Mi}, \qquad [20]$$

where S_{Mi} is the S value for the multicast for partition i. So the total cost of the entire multicast, denoted as $cost_P$, is defined as:

$$Cost_P = \sum_i S_{Mi} (Nq + N + data_size). \qquad [21]$$

5.3.1 Analytical Results

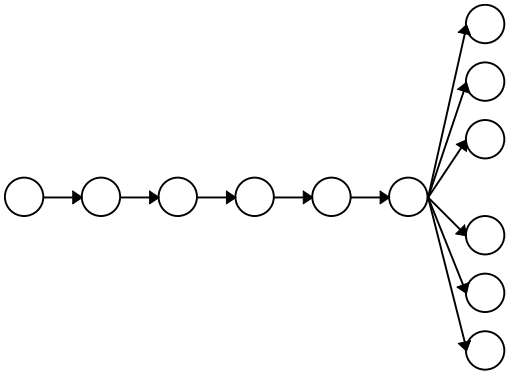

Figure 40: Late Split Topology

Case 1: Late Split

Figure 40 shows a topology where the split occurs late and most of the path from the sender to the recipients is shared. Specifically, only the final element of the path is not shared among all recipients. It represents an efficient multicast topology. For this case,

$S_M = 11$, $S_U = 36$, $S_{Mi} = 7$, for $i = 1..3$.

Efficiency = 1 - 11/36 = 0.69.

$Cost_P = 3\ (18*7) = 378$.

$Cost_M = 11 * (6 * 6) = 396$.

$Cost_U = 36 * (2 * 6) = 432$.

Recall that the cost for the partition method is $(N*q + N + data_size)\ S_{Mi}$, which is $18S_{Mi}$. Since the partitions in the late split and early split examples contain the same number of nodes and all nodes have the same path lengths, the cost of each partition will be the same. Since all partitions have the same cost (except in the combined local and remote topology, Figure 43), the cost for only one partition is calculated and the total cost for all partitions is obtained by multiplying the cost of the partition by the number of partitions, 3. For the causal order multicast method, the overhead from the vectors is N^2, which is 36 in all of the examples. For the delta causal multicast method, the overhead from the vectors is $2N$, which is 12.

The multicast topology in this example is fairly efficient, almost 70%. The cost of all three methods is fairly close in this case, but the partition method has the least cost, with the causal order multicast method second, and the delta causal multicast method being the worst. The delta causal multicast method is the most expensive method in this case because the efficiency of the topology is high enough so that the cost of unicasting 6 smaller messages with two vectors is greater than sending one large message with 6 vectors. The partition approach has the least cost because it performs multicasts efficiently, while sending smaller messages, only three vectors per message. For two partitions of three elements, $Cost_p = 389$, and for only one partition, $Cost_p = 462$. Note

that using two partitions of three elements each is marginally worse than using three partitions with two elements each.

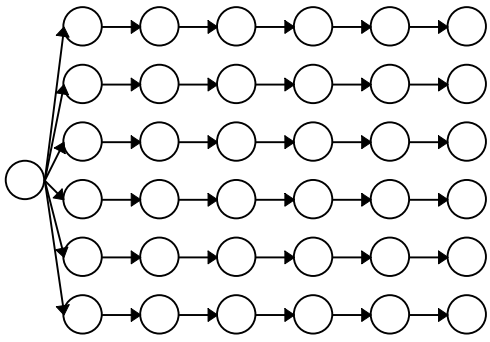

Figure 41: Early Split Topology (worst case scenario)

Case 2: Early Split

Figure 41 shows a multicast topology where elements split as early as possible. No component of the path to any recipient is shared. This is a worst case scenario with an efficiency of 0%. For this case,

$S_M = 36$, $S_U = 36$, $S_{Mi} = 12$, for $i = 1..3$.

Efficiency = 1 - 36/36 = 0.

$Cost_P = 3\ (18 * 12) = 432$.

$Cost_M = 36 * (6 * 6) = 1296.$

$Cost_U = 36 * (2 * 6) = 432.$

This is a worst case example in which the multicast topology is the same as a series of unicasts, since no path component whatsoever is shared (hence, an efficiency of 0%). In this example, the partition method is tied with the delta causal multicast method for having the least cost, with the multicast method being about three times worse. With the low efficiency, sending an entire matrix with every message becomes very costly, resulting in the high overhead of the multicast method. The delta causal multicast method sends each recipient a separate message sharing no information and is well suited for a topology where the efficiency of the topology is low.

The partition method has the same cost as the delta causal multicast method. While it attempts to utilize any underlying multicasts, if there is no inherent path sharing in the topology, there is nothing it can exploit. The relatively low overhead coupled with small partition sizes allows this method to be no more expensive than the delta causal multicast method in this case. While it is designed to exploit underlying multicasting and have a lower cost than the delta causal multicast method, it is not a guaranteed performance, as the following example demonstrates. In a topology similar to Figure 41 in which each path has a length of seven rather than six (because a node is added to each path increasing the total number of nodes from 37 to 43):

$Cost_p = 756.$

$Cost_M = 1512$.

$Cost_U = 504$.

The early split topology is a poor choice to use for any type of multicasting. It is unlikely that this type of topology with this low of an efficiency would be used in practice in a multicasting environment.

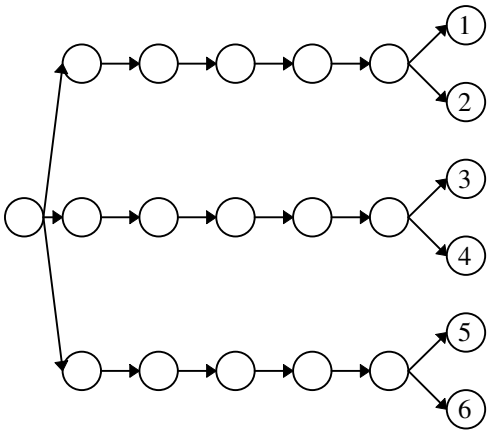

Figure 42: Mixed Split Topology

Case 3: Mixed Split

The mixed split topology combines the characteristics of both the early split and late split topologies. The mixed split topology has an early initial split as well as a second split at the end. Figure 42 shows such a topology where the nodes quickly split into smaller subgroups and the subgroups split into individual recipients at the final stage. This is a

simple model where intuitively it seems the partition method should perform well. In this topology, how partitions are formed has a major impact on the total cost for the partition method. The partitions are constructed efficiently if the nodes in a partition share most of their path, such as partitions ({1, 2}, {3, 4}, and {5, 6}). If the nodes in a partition share little of their path, then the partitions are said to be constructed inefficiently. For this case,

$S_M = 21$, $S_U = 36$, $S_{Mi} = 7$ (if partitioned efficiently, specifically {(1,2), (3,4), (5,6)}), for $i = 1..3$. $S_{Mi} = 12$ (if partitioned inefficiently, e.g., {(1,3), (2,5), (4,6)}), for $i = 1..3$.

Efficiency = 1 - 21/36 = 0.42.

$Cost_{P\ efficient} = 3\ (18 * 7) = 378$.

$Cost_{P\ inefficient} = 3\ (18 * 12) = 848$.

$Cost_M = 21 * (6 * 6) = 756$.

$Cost_U = 36 * (2 * 6) = 432$.

The efficiency of this topology is about 40%. Since this efficiency is not very high, the causal order multicast method does not perform well and the delta causal multicast method is significantly less expensive. If the partitions are constructed efficiently, then the partition method has the least overhead. This is because the partition method takes advantage of the high local efficiency of the partitions due to the high amount of shared paths within each partition, even if the overall efficiency of the multicast topology is

fairly low. However, a poor choice of elements for a partition results in a cost that is even greater than the multicast method. The rest of the examples in this section only consider efficient partitions. Thus the partition method has a cost that is less than the other two methods.

Similarly, for the same topology but with path lengths increased to seven (by adding an additional node in the three middle paths in the graph), the cost of the partition method is still cheaper at 432, with the delta causal multicast at 504 and the causal order multicast method at 864. If the path lengths are increased to 20, the cost of the partition method is 702, the delta causal multicast method is 864 and the causal order multicast method is 2268.

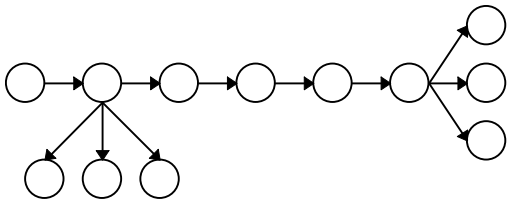

Figure 43: Combined Local and Remote Topology

Case 4: Combined Local and Remote

Figure 43 shows a topology in which there are two types of recipient nodes: local and remote. The local nodes are relatively "close" to the sender and the remote nodes are

relatively "far" from the sender. There are two partitions in this example, one consisting of the three "local" nodes that are two hops away from the sender, and the other consisting of the three "remote" nodes that are six hops away from the sender. Such a situation arises when messages are multicast to several distinct clusters of nodes. For this case,

$S_M = 11$, $S_U = 24$, $S_{M1} = 4$ (local group), $S_{M2} = 8$ (remote group).

$Efficiency = 1 - 11/24 = 0.54$.

$Cost_P = 4*24 + 8*24 = 288$.

$Cost_M = 11 * (6 * 6) = 396$.

$Cost_U = 24 (6 * 2) = 288$.

In this example, the multicast method has considerably more overhead, while the delta causal multicast and partition methods have identical overheads. For the same topology where the remote nodes' path length is 8 (by introducing one additional node in the middle of the topology), the efficiency is approximately 60%, and the costs for partition, causal order multicast and delta casual multicast methods, are 336, 469, and 360, respectively. Similarly, if the path length is 20, the efficiency is approximately 62%, and the costs for the partition, causal order multicast, and delta causal multicast methods are 624, 900, and 792, respectively.

Such topologies, where there are clusters of nodes that share most of their path, are well suited for the partition method, since the nodes can easily be grouped into partitions, where all nodes in a partition share almost every component of their path. While the overall efficiency of the topology is not that high, the partition method can utilize the relatively high local efficiency of the nodes within each partition.

Table 1 summarizes the total costs for these sample topologies presented in the four cases using the three different methods for *data_size* equal to zero. The table also includes values for the combined local and remote topology for different values of the path length of the remote cluster. The first column represents the topology name with the path length in parentheses. The second, third and fourth columns represent the total cost for the delta causal multicast, causal order multicast, and partition methods, respectively. The total cost of a multicast is directly related to the efficiency of the topology and the size of the data. We observe that the partition method consistently performs better than the other methods in most cases, with the exception of the early split topology. Note however, that that topology is poorly suited for multicasting. The performance of the partition method in the combined local and remote topology improves as the length of the remote path increases, since the efficiency increases with the path length. In general, the delta causal multicast method is second, and the causal order multicast method is third with respect to total cost.

Table 2 and Table 3 summarize the costs when *data_size* equals 100 and 1000, respectively, in order to model the total cost for small and large message. As *data_size*

increases, the difference between the least and second least most expensive methods decreases. Even though the multicast method performs well for certain topologies, while for others the delta causal order method performs well, the partition method consistently has a low overhead and generally has the least overhead. In the cases in which it does not have the least overhead, its performance is comparable to the method with the least overhead. Therefore, the partition method has good overall performance for differing topologies.

Table 1: Costs for the Sample Topologies for data_size Equal to Zero.

Topology	**$Cost_U$**	**$Cost_M$**	**$Cost_P$**
Late Split	432	396	378
Early Split (6)	432	1296	432
Early Split (7)	504	1512	756
Mixed Split (6)	432	756	378
Mixed Split (7)	504	864	432
Mixed Split (20)	1440	2268	1134
Combined Local/Remote (5)	252	360	264
Combined Local/Remote (6)	288	396	288
Combined Local/Remote (8)	360	469	336
Combined Local/Remote (10)	432	540	384
Combined Local/Remote (20)	792	900	624

Table 2: Costs for the Sample Topologies for data_size Equal to 100

Topology	**$Cost_U$**	**$Cost_M$**	**$Cost_P$**
Late Split	4096	1496	2478
Early Split	4032	4896	6048
Mixed Split (6)	4032	2856	2478
Mixed Split (20)	8568	13440	7434
Combined Local/Remote (6)	2688	1496	1488
Combined Local/Remote (20)	7392	3400	3224

Table 3: Costs for the Sample Topologies for data_size Equal to 1000

Topology	**$Cost_U$**	**$Cost_M$**	**$Cost_P$**
Late Split	36432	11396	21378
Early Split	36432	37296	54648
Mixed Split (6)	36432	21756	21378
Mixed Split (20)	121440	65268	64134
Combined Local/Remote (6)	24288	11396	12288
Combined Local/Remote (20)	66792	25900	26624

5.3.2 Partition Size

There are various ways to form partitions depending upon the partition size (number of elements in a partition) and what elements are selected for a partition. We now examine the effect that partition size has on the total cost for the partition method.

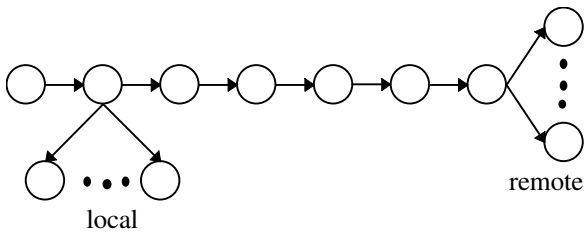

Figure 44: Changing Partition Sizes

The topology from Figure 44 consists of two clusters, a remote cluster in which each recipient has a path length of 8, and a local cluster in which each recipient has a path length of two. First we assume that each local cluster consists of 6 recipients. Without mixing the local and remote recipients, there are 4 different sizes for the partitions: 6, 3, 2, and 1 elements. The respective costs for these partition sizes are 1680, 1344, 1296, 1140.

Now if we assume that each cluster consists of 12 recipients, then there are 6 different sizes for the partitions: 12, 6, 4, 3, 2 and 1 elements. The respective costs for these partition sizes are 9984, 6720, 5760, 5376, 5184, and 5760.

Figure 45 is a plot of the partition size versus the total cost for the partition method for several different values for the number of destinations. With the exception of when there is only one element in a partition, the cost increases as the number of elements in a partition increase. This indicates that the partition method is most efficient when there are a small number of elements in a partition.

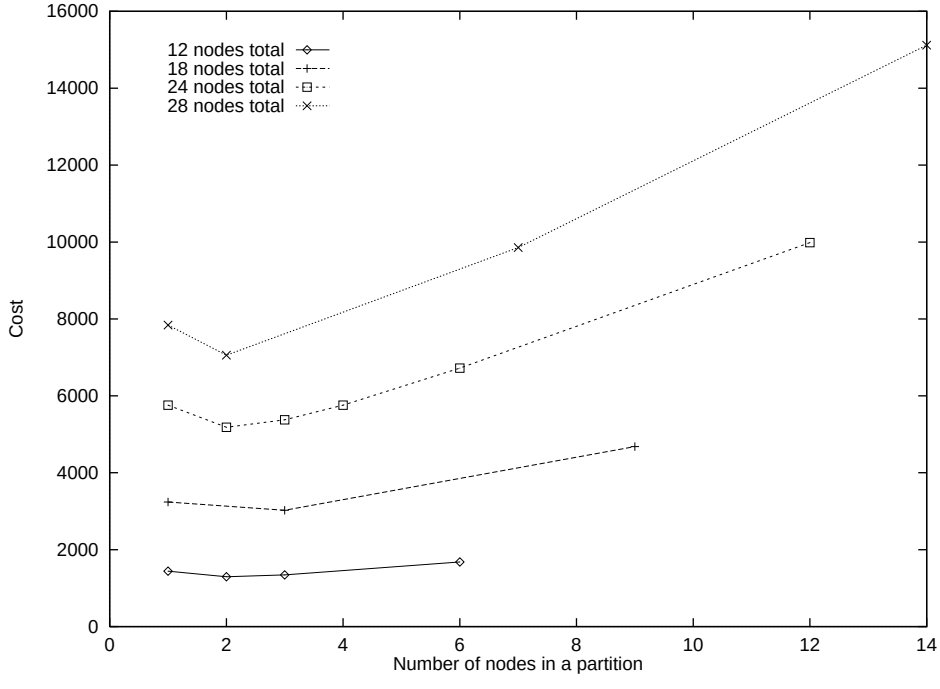

Figure 45: Number of Elements in a Partition vs. Cost

5.4 Summary

ATM networks are high-speed, connection-oriented, cell-switched networks that can provide a guaranteed quality of service and are designed to be used for both local and wide area networks. Because of their high-speed and QOS guarantees, ATM networks are becoming popular in multimedia applications. Causal multicasting algorithms are important to multimedia applications because they guarantee that messages that are multicast to multiple recipients are delivered in a consistent order that preserves cause and effect relationships. For example, causal multicasting can be used for a conference call application that involve recipients at several remote locations. In such an application, the cause and effect relation between messages should be preserved while delivering the data within real-time constraints.

In this chapter we developed a new algorithm to support causal multicasting in ATM networks called the partition method. It exploits the multicast ability of the communication network and typically has a lower cost than the causal order multicast method and the delta causal multicast method presented in the previous chapter.

Utilizing the framework for evaluating the causal multicast algorithms developed in Chapter IV, we analyzed the performance of the three causal multicast methods with respect to their overhead cost for four distinct network topologies. Since the overhead of these methods depends upon the underlying topology of the multicast network, four different sample topologies were used to represent various routing configurations and

efficiencies. While the overhead cost of the three methods is sensitive to the underlying multicast topology, the partition method is also sensitive to the nodes that comprise of each partition and the number of nodes in a partition. We analyzed the performance of the partition method with respect to the partition size, too.

In general, our analysis indicates that if the multicast topology is quite efficient (>65%), the causal multicast method has the least overhead. However, the overhead of sending a matrix of size N^2 with each message quickly increases the total cost as the efficiency decreases. For topologies with lower efficiencies (<40%), the delta-causal ordering method was found to perform best since its message overhead of $2N$ compensates for the lower efficiency. The partition method is well suited for multicast topologies which have distinct groups and the sites in each group share most of their paths with each other. The combined local and remote topology represents such a situation which might arise frequently in ATM networks, since these networks can serve as both a local and wide area networks (LANs and WANs). The partition method performs well with this topology since it exploits the local efficiency of the clusters even when the overall efficiency of the network is not very high.

The partition method is efficient, supports multicast, and in general has a low cost. Its overhead is least when the nodes in a partition share most of their paths, and the number of nodes in a partition is small. The cost equations derived in the developed framework

can be used to measure the cost of causal multicast algorithms and compare them against each other.

6. Conclusions

Chapter VI

Conclusions and Directions for Future Research

This dissertation presented methods for supporting the transmission of multimedia data at the network and operating system level. Due to the availability of faster, cheaper networks and related technology, multimedia is gaining widespread use and is becoming popular. Unlike traditional data, multimedia data is isochronous, or time-sensitive, and imposes strict requirements on the underlying communication network and operating systems in terms of bandwidth, end-to-end delay, and delay variance or delay-jitter. In order to satisfy these requirements, a fast network that provides guarantees on Quality of Service can be employed. However, an alternative to purchasing new hardware technology is to adapt current networking hardware to support multimedia data which is real-time in nature. Ethernet is a very popular local area network that has a wide installation base, but provides no support for real-time data. Extending an Ethernet to support both real-time and non-real-time data was one area explored in this dissertation.

In multimedia applications, such as a conference call, the data should be delivered so that the cause and effect relations between messages from different participants in the

conference is maintained. High-speed, multistage networks can introduce non-uniform delays to messages and cause messages to arrive out of sequence or out of causal order. The service of causal order message delivery should be provided by a layer above the network, typically in the operating systems layer. In order to maintain causal order, control information must be added to each message to allow causal relations to be detected. Another aspect of multimedia support that is presented in this dissertation involves providing causal order delivery of messages for multimedia data with a low overhead.

Applications like conference calls are multicasting in nature since each participant "speaks" to potentially many other participants. The problem of providing causal order delivery of messages becomes more complex in a multicasting environment. Since a multicast message has multiple recipients, control information attached to a multicast message must be appropriate for all of the recipients of the multicast. The cost of sending a message with a causal multicast algorithm is affected by the amount of control information attached to each message as well as by the efficiency of the route of the message through the network. This dissertation developed a framework to evaluate the cost of causal multicast algorithms and used that framework to compare the cost of three different causal multicast algorithms, including one which was developed in this dissertation.

This dissertation focused on three main areas. The first, priority Ethernet, proposed three methods to allow real-time and non-real time data to efficiently share an Ethernet.

The second, real-time causal order, proposed an algorithm to ensure that causal order relations between multimedia messages with real-time constraints are preserved upon delivery. The third, causal multicast, focused on preserving causal order in a multicasting environment. A framework was proposed to analyze the efficiency of a multicast, based on the underlying communication topology and the causal multicast algorithm used to transmit the data. In addition, an efficient causal multicast algorithm was proposed. The research contributions to each of these areas are summarized below.

6.1 Summary of Contributions

6.1.1 Priority Ethernet

In Chapter II, we presented three methods to implement Priority Ethernet protocol. The first protocol, called the synchronous method, virtually eliminates all collisions by imposing a round of priority information exchange during which each site broadcasts their priority to all other sites on the Ethernet. Due to the elimination of collisions, the Ethernet sustains a much higher throughput with the synchronous method than with just the CSMA/CD protocol. The second protocol, called the modified synchronous method, relaxes the constraint that all sites must broadcast their priority even if they have no message to send. This method reduces the time wasted in the priority information exchange rounds by sites that have no message to send. Finally, the third protocol, the asynchronous method, eliminates the priority information exchange phase; instead, it allows sites to broadcast their priority information as messages arrive. By eliminating the

priority information exchange rounds, this method allows high priority data to have a low response time in heterogeneous traffic, which permits higher sustained throughputs than with just CSMA/CD.

A qualitative analysis was presented to describe the types of traffic in which each method will perform best. A simulation study was conducted, which simulated all three methods as well as an Ethernet with CSMA/CD only. The simulations were run with data from both non-bursty (Poisson) and bursty sources and the response time for the proposed methods was compared to that of the plain Ethernet. The simulation showed that the synchronous method has very good performance, especially in the presence of bursty traffic. The modified synchronous method has a better performance than that of the plain Ethernet, although for uniform traffic it has slightly lower performance than the synchronous method. The asynchronous method's performance was adequate, but not a large improvement over plain Ethernet. However, the asynchronous method's performance improves under a non-uniform traffic load.

6.1.2 Causal Order

In Chapter III, we presented an algorithm to ensure causal order for multimedia messages. By focusing on causal order violations due to the triangle inequality, the algorithm has a low message overhead, $O(2N)$, as opposed to $O(N^2)$ which other methods impose. The algorithm enforces delta causal order, which is a form of causal order restricted to a certain time interval. This is applicable to multimedia data which is

real-time in nature and must be delivered by a deadline or be discarded. A proof of correctness was presented and we also showed that 2 vectors of length *N* are both necessary and sufficient to enforce delta causal order. In addition, if the data is homogeneous, i.e., all data has the same time constraints or lifespan, then every message that arrives at its destination before its deadline will be delivered rather than discarded.

6.1.3 Causal Multicast

Chapters IV and V focused on causal multicasting. In Chapter IV, we presented a framework to characterize and evaluate the cost of a multicast given the multicast algorithm and the underlying network topology. We showed that both the network topology as well as the data that is sent influence the cost of a multicast. We used the framework to characterize the cost of two different causal multicast algorithms and presented a criterion to determine which method has the least cost, given the size of the data to transmit and the efficiency of the underlying topology. By utilizing this criterion, the lower cost algorithm can be used resulting in higher network throughput.

In Chapter V, we presented the partition method, a low overhead causal order algorithm that supports multicasting. We compared its performance and the performance of the causal order multicast method and the delta causal order multicast method, for several different sample topologies and showed that the Partition Method has low cost and high efficiency and is well suited for most topologies. We analyzed the cost of the partition

method for various partition sizes and showed that small, highly efficient partitions result in the lowest cost.

6.2 Future Directions

Future research in Priority Ethernet involves investigating the performance tradeoffs and technological demands in placing functions in various layers in the layered architecture (i.e., protocol structuring and placement issues). In addition, the protocols should be expanded to support multiple physical Ethernet networks. Since the protocol performance is traffic dependent, the performance for different traffic models should be explored as well as possible improvements for handling bursty traffic, such as, allowing each site to transmit more than one packet per round.

Future work for the causal order algorithm involves experimental simulation and providing fault tolerance.

Future work for causal multicasting includes a simulation study of the performance of the three methods. Since the performance of the methods is topologically dependent, it will be necessary to experimentally determine what types of multicast topologies will be commonly used by observing real networks. Finally, a hybrid approach that switches among the three different methods based on the topology and the size of the data may be the best method and should be explored.

Other future directions include work on an efficient routing algorithm that selects the most efficient multicast topology based on the cost of the network topology and the cost of the algorithm and experimental simulation or implementation on an ATM network.

Bibliography

1 Frank Adelstein and Mukesh Singhal, "*Real-Time Causal Message Ordering in Multimedia Systems*", to appear in the Proceedings of the 15th International Conference on Distributed Computing Systems, Vancouver, June 1995. (Also available as Technical Report, OSU-CISRC-3/94-TR14, The Ohio State University, March 1994.)

2 Khaled Amer, Ken Christiansen, Tora Toher, "*Experiments With Client/Server Multimedia On Token Ring*", Proceedings of the 18th Conference on Local Area Networks, 1993.

3 Khaled Amer, Ken Christiansen, Tora Toher, "*Multimedia Applications on IBM 16-bit Token-Ring LANs"* IBM Personal Systems Journal, April, 1993

4 Dave Andrews, "*High Stakes for High-Speed Ethernet"*, Byte, October, 1993, pp. 22 - 23.

5 Rajiv Arora and Alexander Albicki, "*Prioritized Access in CSMA Networks: A Node Partitioning Approach*", Proceedings of the 13th Conference on Local Computer Netowrks, Minneapolis, Minnesota, October, 1988., pp, 275 - 280.

6 Jean-Yves Le Boudec, "*The Asynchronous Transfer Mode: a tutorial*", Computer Networks and ISDN Systems, Vol. 24, pp. 279 - 309, 1992.

7 Kenneth Birman, André Schiper, and Pat Stephenson, "*Lightweight Causal and Atomic Group Multicast*," ACM Transactions on Computer Systems, Vol. 9, No. 3, August 1991, pp. 272 - 314.

8 Gagan L. Choudhury and Stephen S. Rappaport, "*Priority Access Schemes Using CSMA-CD*", IEEE Transactions On Communications, Vol. Com-33, No. 7, July 1985, pp. 620 - 626.

9 Nim Cheung, "*The Infrastructure for Gigabit Computer Networks,*" IEEE Communications Magazine, Vol. 30, No. 4, April 1992, pp. 60 - 68.

10 Nick Chinai, "*A Preview of Fast* Ethernet", Smart-Net, pp. 71 - 75, Spring/Summer 1993.

11 Douglas Comer, "Internetworking With TCP/IP: Principles, Protocols and Architecture", Prentice-Hall, 1988.

12 Computer, Special issue on Multimedia Information Systems, Vol. 24, No. 10, October 1991.

13 The Computer Journal, Special issue on multimedia, Vol. 36, No. 1, 1993.

14 Jean-Pierre Coudreuse, "*Communications Using ATM*", Encyclopedia of Telecommunications, Vol. 4, 123 - 164, 1992.

15 John Matthew Simon Doar, "*Multicast in the Asynchronous Transfer Mode Environment*," Ph.D. Thesis, St. John's College, University of Cambridge, January 1993.

16 A. Elsaadany, M. Singhal, M. Liu, "*Multimedia on Local Area Networks*", in Proceedings of the 19th Symposium on Local Area Networks, October, 1994.

17 Domenico Ferrari, "*Client Requirements for Real-Time Communication Services*", IEEE Communications Magazine, November, 1990, pp. 65 - 72.

18 Wolfgang Fischer, Kathleen Meier-Hellstern, "*The Markov-modulated Poisson process (MMPP) cookbook*", Performance Evaluation, North-Holland, 18, 1992, pp. 140 - 171.

19 R. Gusella, S. Zatti, "*The Accuracy of the Clock Synchronization Achieved by TEMPO in Berkeley UNIX 4.3BSD*," IEEE Transactions on Software Engineering, Vol. 15, No. 7, July 1989, pp. 847 - 853.

20 IEEE Journal On Selected Areas In Communication, Special issue on multimedia, Vol. 8, No. 3, April 1990.

21 IEEE Transactions On Knowledge And Data Engineering, Special issue on multimedia, Vol. 5, No. 4, August 1993.

22 The Institute of Electrical and Electronics Engineers, "*Carrier Sense Multiple Access with Collision Detections (CSMA/CD) Access Method and Physical Layer Specifications*", American National Standard ANSI/IEEE Std. 802.4-1985.

23 International Standards Organization, "*Basic Reference Model for Open System Interconnection*", ISO 7498, 1984.

24 Chong-Kwon Kim and Tony T. Lee, "*Call Scheduling Algorithms in a Multicast Switch,*" IEEE Transactions on Communications, Vol. 40, No. 3, March 1992, pp. 625 - 635.

25 Leslie Lamport, "*Time, Clocks, and the Ordering of Events in a Distributed System*", International Standards Organization, "Basic Reference Model for Open System Interconnections", ISO 7498, 1984.

26 J. Bryan Lyles and Daniel C. Swinehart, "*The Emerging Gigabit Environment and the Role of Local ATM*," IEEE Communications Magazine, April 1992.

27 M. Macedonia and D. Brutzman, "*MBone Provides Audio and Video Across the Internet*", IEEE Computer, Vol. 27, No. 4, April 1994, pp. 30 - 36.

28 E. Mafla, Bharat Bhargava, "*Communications Facilities for Distributed Transaction-Processing Systems*", IEEE Computer, August, 1991, pp. 61 - 66.

29 F. Mattern, "*Virtual Time and Global States of Distributed Systems*", Proceedings of the International Workshop on Parallel and Distributed Algorithms, pp. 215 - 226, Amsterdam, 1989.

30 R. Metcalfe and D. Boggs, "*Ethernet: Distributed Packet Switching for Local Computer Networks*", Communications of the ACM, July, 1976, Vol. 19, No. 7, pp. 395 - 404.

31 David Mills, "*Internet Time Synchronization: The Network Time Protocol,"* IEEE Transactions on Communications, Vol. 39, No. 10, October 1991, pp. 1482 - 2493.

32 Daniel Minoli, "*Isochronous Ethernet: Poised for Launch*", Network Computing, pp. 156 - 161, August 1993.

33 Achour Mostefaoui, Michel Raynal, "*Causal Multicasts in Overlapping Groups: Towards a Low Cost Approach*," IEEE International Conference on Future Trends of Distributed Computing Systems, Lisboa, September, 1993.

34 A.D. Narasimhalu and S. Christodoulakis, "*Multimedia Information Systems: The Unfolding Reality (Guest Editors' Introduction)*", Computer, October, 1991, Vol. 24, No. 10, pp. 6 - 8.

35 Ciro A. Noronha Jr. and Fouad A. Tobagi, "*Optimum Routing of Multicast Streams*," in the Proceedings of the 13th IEEE Infocom, 1994, pp. 865 - 873.

36 Larry Peterson, Nick Buchholz, and Richard Schlichting, "*Preserving and Using Context Information in Interprocess Communication*," ACM Transactions on Computer Systems, Vol. 7, No. 3, August 1989, pp. 217 - 246.

37 M. de Prycker, *Asynchronous Transfer* Mode, Ellis Howood Ltd., 1991.

38 A. Schiper, J. Eggli, A. Sandoz, "*A New Algorithm to Implement Causal Ordering*", In Proceedings of the International Workshop on Distributed Algorithms, 1989, in Lecture Notes in Computer Science 392, Springer-Verlag, New York, pp. 219-232.

39 Herb Schwetman, "*CSIM Reference Manual (Revision 16)*", MCC Technical Report Number: ACA-ST-252-87, Rev. 16, June 1992.

40 Nachum Shacham and James S. Meditch, "*An Algorithm For Optimal Multicast of Multimedia Streams*," in the Proceedings of the 13th IEEE Infocom, 1994, pp. 856 - 864.

41 Sheldon H. Shen, Howard T. Liu, "*Dynamic CSMA/CD With A Priority Scheme*", INFOCOM '86, IEEE, 1986, pp. 258 - 263.

42 M. Singhal, A. Kshemkalyani, "*An Efficient Implementation of Vector Clocks*", Information Processing Letters, 43, pp. 47-52, August 1992.

43 William Stallings, Data and Computer Communications, MacMillan, 1988, pp. 337 - 351.

44 Sun Microsystems, Inc., *"LANCE 10Mbit/sec Ethernet interface"*, June, 1992. (From the online manual page *le(4).*)

45 Andrew S. Tanenbaum, Computer Networks", Prentice-Hall, 1989.

46 Fouad A. Tobagi, "*Carrier Sense Multiple Access with Message-Based Priority Functions*", IEEE Transactions On Communications, Vol. COM-30, No. 1, January 1982, pp. 185 - 200.

47 Fourad A. Tobagi, "*Multimedia: The Challenge Behind the Vision*", Data Communications, January, 1993, pp. 61 - 66.

48 Jonathan Turner, "*An Optimal Nonblocking Multicast Virtual Circuit Switch*," in the Proceedings of the 13th IEEE Infocom, 1994, pp. 298 - 305.

49 Raj Yavatkar, "MCP: A Protocol For Coordination and Temporal Synchronization in Multimedia Collaborative Applications," In the Proceedings of the 12th International Conference on Distributed Computing Systems, 1992, pp. 606 - 613.