INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI®

Towards a Better Architecture for Reliable Multicast Protocols

Aiman Hanna

A Thesis

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Computer Science at

Concordia University

Montreal, Quebec, Canada

March 2000

ABSTRACT

Towards a Better Architecture for Reliable Multicast Protocols

Aiman Hanna

The definition of reliable multicast has varied widely, going from requirements for absolute data reliability, strong group membership control, strong ordering guarantees, multiple senders, and small group sizes, to protocols with weaker data reliability, little or no group membership control, no ordering guarantees, single senders, and large to very large group sizes. The first group tends to have a flat error-recovery structure, and the second group tends to use a hierarchical error-recovery structure. A review of 8 representative "reliable multicast" protocols, each with a different definition of reliability, has identified a number of common characteristics, and distinct differences. As a step towards the definition of a protocol that is applicable to a wider range of applications, a new architecture is proposed in this thesis. It combines a central core operating as a fully-reliable, many-to-many protocol, providing communication among a group of" significant receivers", with a hierarchical, scalable, one-to-many protocol, providing distribution of the core data to a set of "simple receivers". The protocol is designed as a set of mechanisms to be invoked, or ignored, as needed. A mapping of the architecture to the representative protocols is presented.

To My Wonderful Parents

Mr. Latif Hanna and Mrs. Hoda Hanna

And My Sweetest Wife

Athena.

Acknowledgement

"Unless the LORD builds the house, its builders labor in vain. Unless the LORD watches over the city, the watchmen stand guard in vain." – Holy Bible, Psalm 127:1.

It is with deep appreciation that I acknowledge the immense help of my supervisor, Dr. J.W. Atwood. I am so fortunate to work with an expert like him in the field of Networking and Data Communication, as well as a truly pleasant person in life. Although we have faced many challenges and obstructions throughout our research years, all I remember is just good memories. Indeed, I have learned enormously from him for both my career and life. It is true that "thanks" is just a small word, however, I will for life hold as much appreciation as this nice little word may indicate.

A special thanks to members of the XTP Forum, especially Tim Strayer and John Fenton, for providing us with Sandia XTP and Mentat XTP. Their contribution is greatly appreciated.

I would like to thank my parents, Mr. Latif Hanna and Mrs. Hoda Roufaiel for their endless love and support to me since my first moment in life. Their love, support and infinite belief in me have been a great reason behind all my achievements.

I would like to thank my soul mate, my sweetest wife, Athena. She has always been my inspiration. She has shared my tiring years to complete this work. For her, I dedicate this thesis.

I would also like to thank my brother, Mr. Ashraf Hanna for his constant motivation, and tremendous belief in me. He has always played a part in my success.

Last but not least, I would like to express my deep appreciation to the faculty, staff, system analyst team and my colleagues at the Department of Computer Science at Concordia University. I am so thrilled to be around such scientists and professionals like them. In the end, it is this that has made it all worthwhile.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In recent times, very high-speed communication networks are considered critical elements of global information systems for applications such as defense and intelligence, supercomputing and education, and medical imaging. Many upcoming applications such as distributed parallel processing, white boards, network conferencing and remote cooperative work essentially require reliable multi-point communications.

With this increasing amount of communication, the consumption of network resources has become a very critical issue. Achieving multi-point reliable communication via simple point-to-point data transmission is no longer appropriate due to the excessive and inefficient resource consumption. This was a first force towards designing a better network communication technique, referred to as multicasting.

Multicasting has quickly turned to be a very significant subject in network communication. Much cooperative work has been conducted by various agencies and universities to design supportive mechanisms for multicasting; these mechanisms were then referred to as multicast protocols.

Unfortunately, although various multicast protocols currently exist, none of these protocols succeed either to provide all the major multicast features, or to become suitable for all or even a majority of applications. For example, many protocols do not support M-to-N multicast at all. Others support M-to-N multicast, however, on the cost of scalability.

The reason all of these protocols have failed to become suitable for most applications and failed to provide all the major multicast features is mainly due to the way these protocols have been architected. Our motivation in this thesis is to introduce a new architecture for multicast protocols, with which a protocol can

efficiently provide a mixture of all the major multicast functionalities, and hence become suitable for a vast number of applications. Moreover, our concern is not to build another multicast protocol, but to be able to incorporate this architecture into many of the existing protocols, and thus enhance the service provided by these protocols. Yet, new multicast protocols can certainly be built upon this architecture. In either case, we are hopeful that the work presented in this thesis will be a step towards achieving a standard reliable multicast protocol.

1.2 A Roadmap through the Thesis

This chapter, chapter 1, introduces the major techniques of network communication; these are unicasting and multicasting. A brief comparison between the two techniques is given. Additionally, the chapter gives a brief introduction of transport layer protocols and indicates the major known reliable multicast protocols.

Chapter 2, introduces the major entities of a multicast session. The chapter explores some of the major multicasting issues, such as the different modes of multicasting, hardware multicasting, group membership management, and IP multicast routing. Some of the major multicast routing protocols are then examined.

Chapter 3 explores in depth the concept of reliable multicasting. In addition, the chapter examines other major multicasting issues such as scalability, ordering, flow control, late-joining receivers and failure recovery.

Chapter 4 examines the major reliable multicast protocols, which are: the Tree-based Multicast Transport Protocol (TMTP), the Scalable Reliable Multicast (SRM), the Reliable Multicast Transport Protocol (RMTP), the Reliable Adaptive Multicast Protocol (RAMP), the Reliable Multicast Protocol (RMP), the Multicast Transport Protocol (MTP-2), the Local Group based Multicast Protocol (LGMP), and the Xpress Transport Protocol (XTP).

Although various reliable multicast protocols exist, none of them was able to provide all of the major multicast requirements. The way each of these protocols has been structured is the main reason behind this significant limitation. In chapter 5, we present a new architecture for reliable multicast protocols. With this architecture, a protocol should be able to efficiently provide all of the needed multicast requirements at the same time. The incorporation of this architecture into many of the existing protocols is also feasible. Chapter 5 includes a full description of the architecture, as well as these associated issues.

The last chapter, Chapter 6, gives an overview of the work described in this thesis and highlights the needed future work.

1.3 Unicasting and Multicasting

1.3.1 Unicasting

Unicasting is considered the most efficient technique for a point-to-point communication. Whenever two processes wish to communicate with each other, a unicast connection is established between them. The connection can be constructed to be either unidirectional, where the data flows only in one direction, or bi-directional, where the data may flow in either direction. The decision of having the unicast connection to be unidirectional or bi-directional depends on the application requirements. For example, sender-receiver applications with no reliability requirements require only unidirectional connections. However, if both processes need to exchange information, then a bi-directional unicast connection must be established. In either case, with correct configurations of the communication parameters, it is possible to achieve the most efficient usage of network resources. For example, setting the round trip time to a proper value can eliminate the transmission of duplicate packets, hence reducing network traffic and minimizing bandwidth consumption. Moreover, with point-to-point unicasting, many communication requirements such as reliability, error recovery and packet ordering can be easily implemented.

Unfortunately, if the number of communicating processes in a session becomes more than two - referred to as multi-party communication or multi-party data flow session - unicasting does not behave as well any more. For example, for a multi-party of N processes, should a process wish to send a packet to the rest of the group, N-1 duplicates of the packet have to be sent in the N-1 unicast connections between the sender and the receivers. This can be simply thought of as an N times increase of network traffic and bandwidth consumption. As a result, for multi-party communications, a better technique than unicasting should be used: that is Multicasting.

1.3.2 Multicasting

Multicasting is a technique that provides a very powerful mechanism for multi-party communication. Should a process wish to transmit a packet to a number of processes, it needs only to transmit the packet to a multicast address. All other processes can receive the packet by simply listening to that multicast address. That is, with multicasting, sending one single copy of the packet is sufficient for the whole group of members to receive it. This significantly reduces network traffic and bandwidth utilization.

Certainly, this improvement does not come for free. With multicasting, the implementation of many communication requirements, such as reliability, error recovery, congestion control, flow control, scalability and packet ordering becomes more complex. Additional problems that never existed with unicasting, such as implosion, are present with multicasting, hence extra solutions are required. Additionally, the simple above-mentioned scenario for 1-to-N multicast is not sufficient for providing M-to-N multicast. More complex algorithms and techniques are needed to provide such a facility. New protocols and applications are under development worldwide to handle these issues.

1.3 Transport Layer Protocols

The need for reliable multicasting was the drive to designing many new transport layer protocols. The most recognized and broadly used transport protocols, TCP and UDP, fail to provide the needed

services for most multicasting applications. To be precise, TCP does not provide any multicasting features, while UDP provides only unreliable multicasting. For some applications, such as video and audio conferencing, end-to-end reliability may not be needed. The data for such applications is transient. For these applications, reliability can simply be achieved just by skipping over missing segments. The implementation of such applications can be adequately based on UDP.

However, other collaborative applications, such as medical imaging, Jacobson's whiteboard tool (wb), shared text editors, image archiving, detailed image analysis and processing and shared databases, require consistency across multiple views. For such applications, reliability must be provided. The need for these applications resulted in the development of a few transport layer protocols that provide reliable multicasting services, such as: the Xpress Transport Protocol (XTP), the Tree-based Multicast Transport Protocol (TMTP), the Scalable Reliable Multicast (SRM), the Reliable Multicast Transport Protocol (RMTP), the Reliable Adaptive Multicast Protocol (RAMP), the Reliable Multicast Protocol (RMP), the Multicast Transport Protocol (MTP-2), and the Local Group based Multicast Protocol (LGMP).

Chapter 2

Multicasting – The Concept and The Necessity

2.1 Basic Concept of Multicasting

A multicast service can be simply defined as the process in which a single data unit transmitted by a source is received by multiple destinations. The set of participating members in the packet multicast service composes a multicast group. The multicast group members communicate using an intermediate entity called the multicast server, which provides the multicast services to all members [Recommendation X.6]. Not all the members of a multicast group are required to participate in the data transfer. Some members may participate only in management issues such as security and control. Additionally not all the members of a multicast group are required to be active. Active members are those who participate in a particular multicast data transfer operation at a given time instance [Recommendation X.6]. That is, the active set may change during the lifetime of the multicast session. The creation, management and control of a multicast group are usually the responsibility of either one of the members, or just a third party. This entity is always referred to as the group controller.

2.2 Senders and Receivers of a Multicast Session

In a multicast session, members can be classified based on their communication role. A member can be a sender, a receiver or possibly both. If the session provides merely one-to-N multicast then only one member is designated as the sender, while the rest of the members are designated as receivers. In such cases, a receiver member may never send data. However, if the multicast session is a M-to-N then more than one sender may exist. All senders in such communication are allowed to transmit data. Optionally, a sender can also act as a receiver for packets that are sent by other senders, or possibly by itself [Recommendation X.6]. Finally, if the session provides N-to-N multicast, then each member can act as both a sender and a receiver. Such members have the right and capabilities of receiving data as well as sending data.

Although in M-to-N and N-to-N multicast more than one member does have the capability to transmit data, those members may still need to obtain some form of authorization before being able to transmit information. This authorization may be controlled by a specific member, such as the group controller, or by a session level application.

2.3 Multicasting Modes of Data Flow

Data flow refers mainly to the direction in which the data is transferred between two entities. Different modes of data flow exist. A multicast server may provide all possible modes or only a subset of them. The different data flow modes are: one-way data flow, two-way data flow and n-way data flow.

2.3.1 One-way Data Flow

In one-way data flow, the transmission is simplex. The data can only be transmitted from the sender(s) to the receivers. That is, there is no return data path from the receivers to the sender(s). Receivers in this mode are capable only of receiving data. A sender may optionally be capable to receive data sent by other senders, or by itself [Recommendation X.6]. This later case may not be useful, however it is available should an application require it.

A simple case of one-to-N multicast with one-way data flow is shown in figure 1. Figure 2 shows another case of one-way data flow for an M-to-N multicast session.

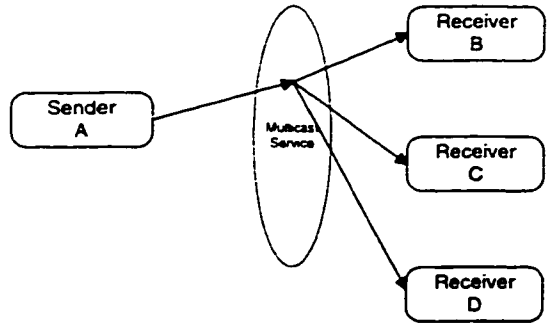

Figure 1: One-way Data Flow in One-to-N Multicasting

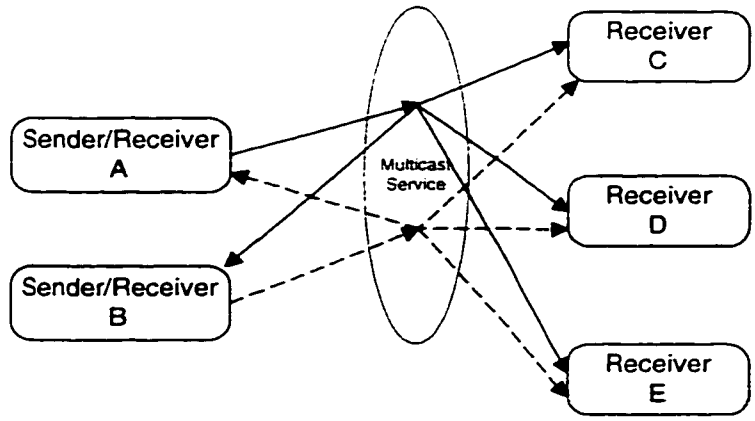

Figure 2: One-way Data Flow in an M-to-N Multicast Session

2.3.2 Two-way Data Flow

In two-way data flow, the transmission is duplex. The data can be transmitted from the sender(s) to the receivers, as well as from the receivers to the sender(s). That is, there is a return data path between each receiver and sender. Receivers in this mode are capable of receiving data, as well as transmitting data to the senders. Data in the return paths, which is sent by a receiver, is transmitted to all senders [Recommendation X.6]. Receiver-to-receiver transmission is not allowed in this mode. A sender may optionally be capable of receiving data sent by other senders, or by itself. With this mode, it is possible that many receivers send data back to a sender in the reverse data paths between them and the sender. This situation where the transmission takes place from many sources to one destination is called concentration. Figure 3 shows a two-way data transfer in a one-to-N multicast session, while figure 4 shows two-way data transfer in an M-to-N multicast session. Notice that the transmission from sender(s) to receivers is a multicast transmission, while the transmission in the reverse direction from receivers to sender(s) is a concentrated transmission.

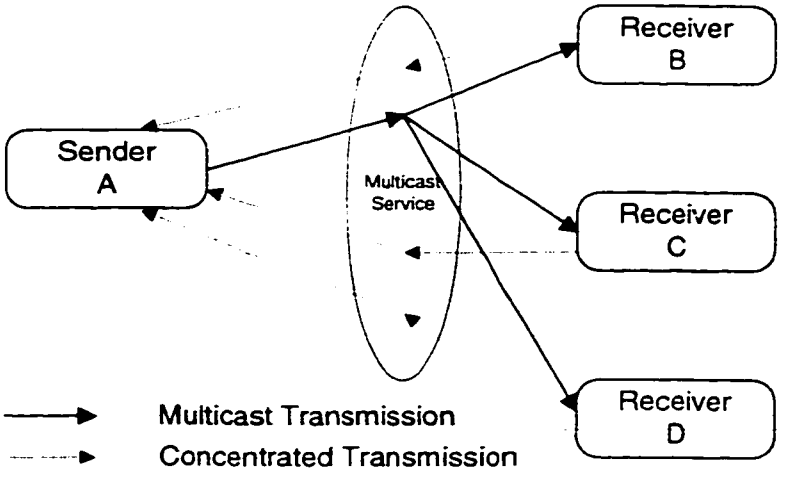

Figure 3: Two-way Data Flow in a One-to-N Multicast Session

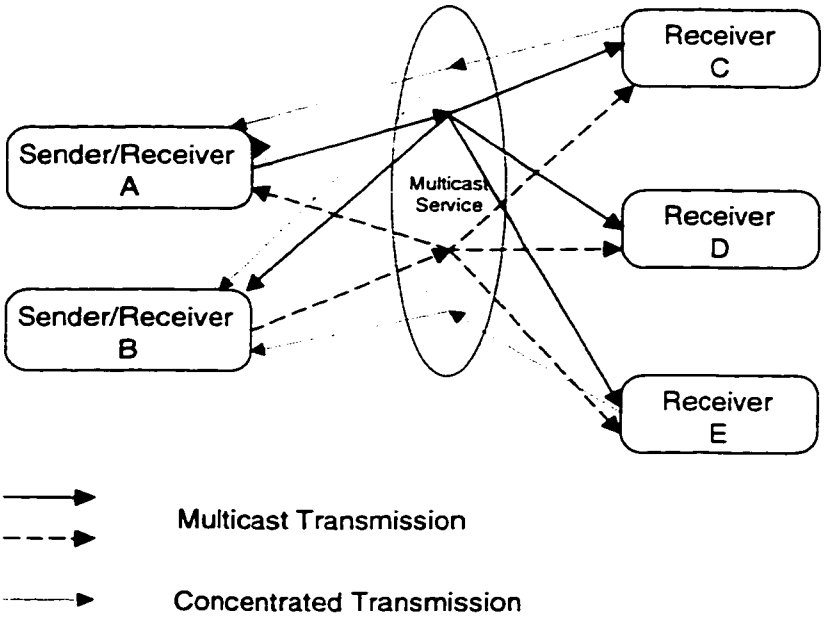

Figure 4: Two-way Data Flow in an M-to-N Multicast Session

2.3.3 n-way Data Flow

In n-way data flow, the transmission is duplex. In addition, all transmissions are multicast. All entities in such communication are capable of transmitting data as well as receiving data. A packet sent by any member is received by all other members, and optionally by itself [Recommendation X.6]. This kind of

data flow simply leads to a complete N-to-N multicasting. Figure 5 shows an example of such communication.

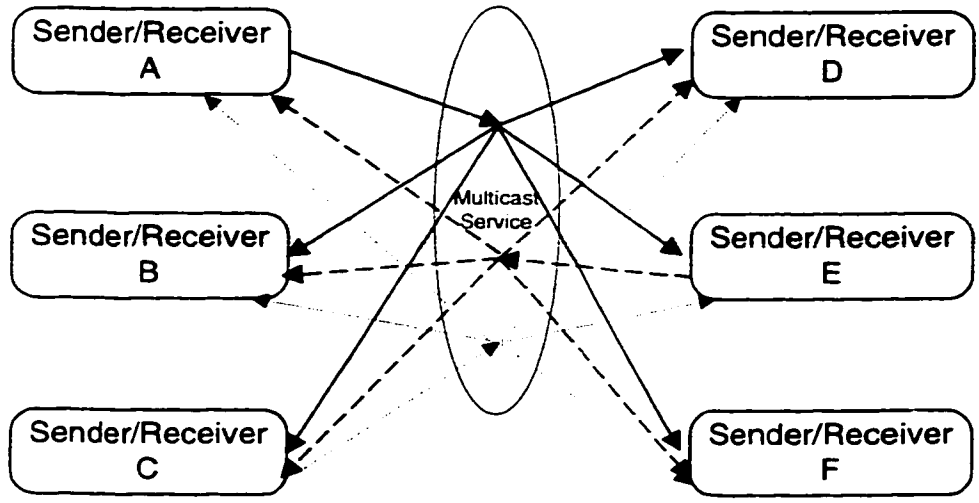

Figure 5: n-way Data Flow, N-to-N Multicasting

2.4 Multicast Model

The multicast model can simply be viewed as all entities in a multicast session along with the logical relations between them. Entities of a multicast session are composed of the different members (senders and receivers), and the multicast server. The multicast server is the entity that provides the multicast services to all the members [Recommendation X.6].

Hence, the multicast server defines the main characteristics of a multicast session. In practice, the multicast server may be a single entity or a set of distributed entities. In addition, the multicast server can be on the same network as the members, or possibly on a different network. If the multicast server is distributed on different networks, then an inter-networking capability is needed to allow these servers to communicate with each other. The server-to-server communication is transparent to the members. For the members, the service is always provided by a single logical entity - the multicast server [Recommendation X.6]. Figures 6 through 8 show examples of these different situations.

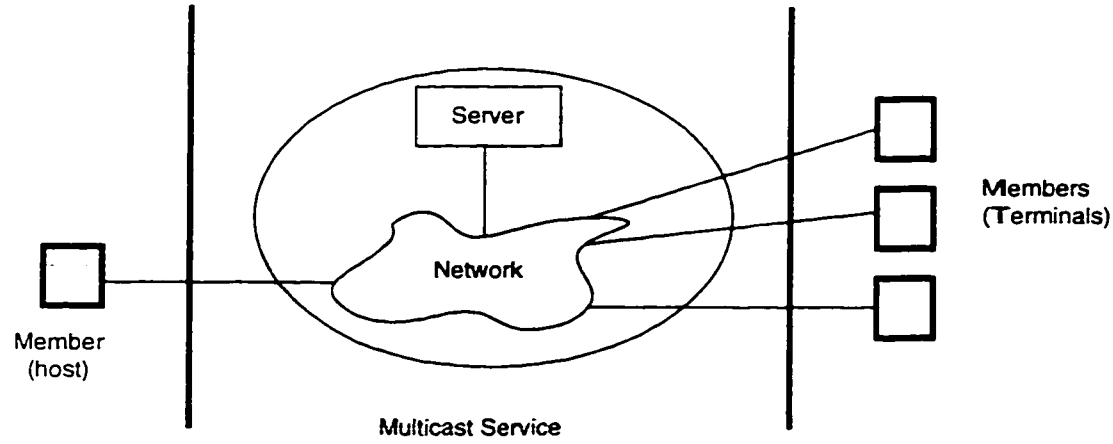

Figure 6: Single Server Outside the Network

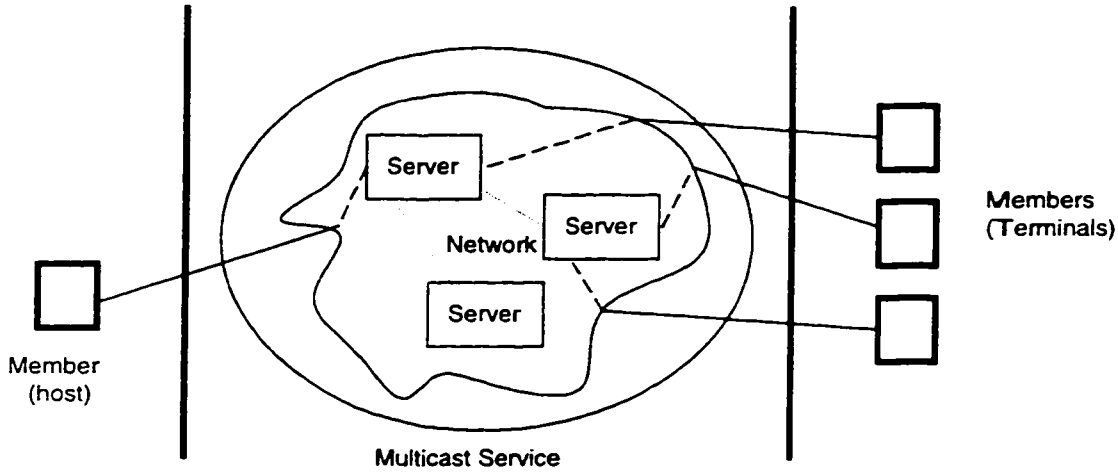

Figure 7: Distributed Server Inside the Network

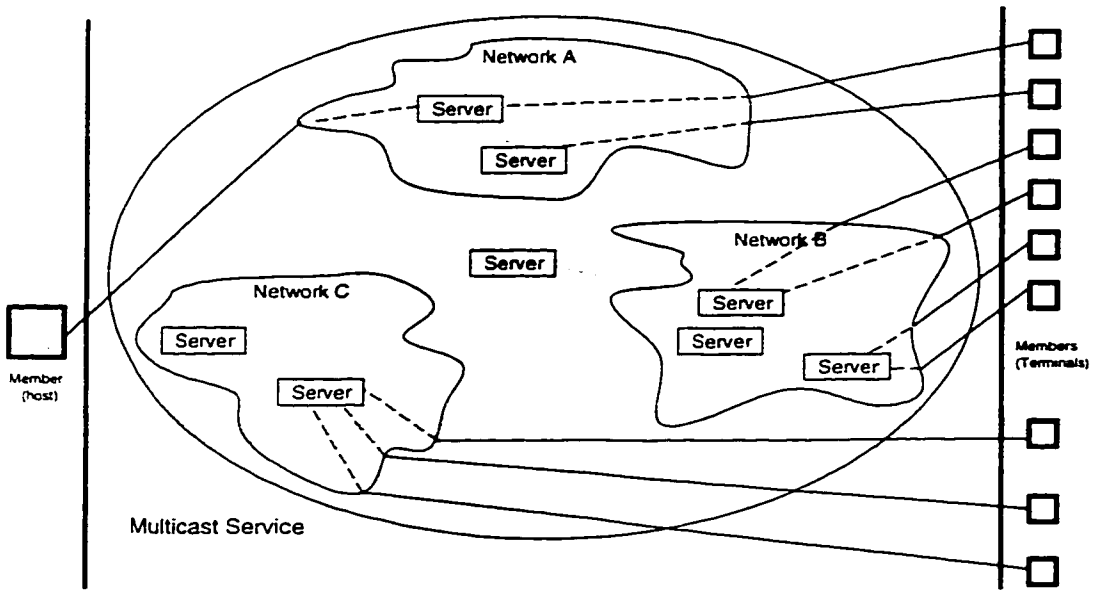

Figure 8: Distributed Server both Inside and Outside of Multiple Networks

2.5 Multicast Group

A multicast group is composed of all members participating in a multicast communication. A multicast group can be static, which means that the members are predefined and unchangeable within the session lifetime, or it can be created and changed dynamically on-line. Members of a multicast group can communicate with each other through multicast calls. A multicast call defines the logical relationship between members for the purpose of transferring data [Recommendation X.6]. Hence, a multicast call for a group of N members may be logically composed of N point-to-point connections, one between each member and the multicast server.

A dynamic multicast group does not have a fixed pre-defined set of members, but members can join or leave the group at any point within the session lifetime. Such a multicast group is called an open group. Open groups are usually set for applications where members are not known beforehand, or when the application needs require such flexibility.

Adding or removing members from a multicast group is controlled by a special entity called the multicast group controller. The group control capabilities can be provided through static administrative means or through dynamic on-line methods [Recommendation X.6]. With static administration, adding or removing members is achieved by sending requests to the administration entity, which performs the operation. With dynamic on-line methods, there is an additional flexibility, as the multicast group controller is capable of adding or removing members using some provided on-line methods. These on-line procedures may be provided by an extension to the standard network management protocols [Recommendation X.6].

A multicast group is identified using a group identifier (ID), which is assigned to it once the group is created. The ID can be designed as the network address of the server plus an additional identifier that is assigned by the server. Alternatively, the ID can be designed as a closed user group interlock code, or simply as a single network address. However, within an X.25 network, the ID is designed as an international closed user group interlock code [Recommendation X.6]. In all cases, the ID must be unique within a network. Within a larger context, the ID can be combined with a network address to provide a globally unique group ID.

Not all members of a multicast session may share all multicast calls. Any set of the members may participate in a specific multicast call in a given time. The set of participating members in a multicast call composes what is called an active group. Figure 9 shows an example of different multicast calls within a multicast session. As shown, there are three active groups, one for each multicast call. Any member can participate in more than one multicast call, hence become a member of more than one active group.

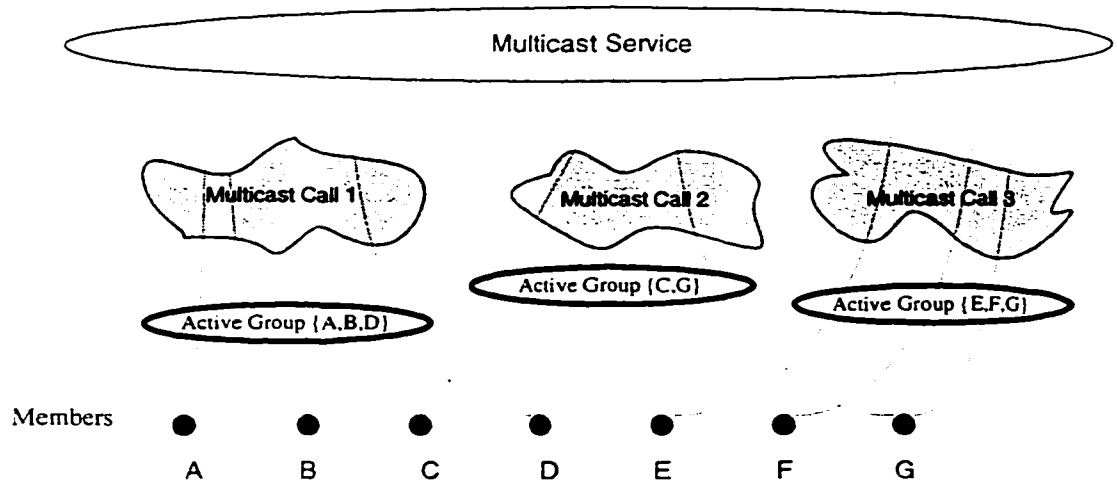

Figure 9: Multicast Calls and Active Groups

2.6 Multicast Member Capabilities

Depending on the functions that it is required to perform, a group member may have one or more capabilities. Not all the capabilities a member has may be realized at all times. When a group member participates in a call, the needed set of its capabilities is used to support its role in this specific instance of the communication. In general, the main capabilities that a member may have are: initiate, send, receive, join, leave and terminate. The initiate capability allows a member to initiate multicast calls. The send capability allows a member to transmit data to other members of the multicast session. In a one-way data flow/one-to-N multicasting, only one member may have this capability. A receive capability allows a member to receive packets transmitted by one or more senders within the multicast session. A join capability allows its holder to request joining an in-progress multicast call, while the leave capability allows a member to disconnect itself from a multicast call. A terminate capability allows its holder to terminate a call in progress [Recommendation X.6].

Additional capabilities may be allowed to members such as, receive join/leave notification, join permission, invite, exclude and control messages. The receive join/leave notification allows a participant to receive a notification whenever a member joins or leaves the multicast call. A member that has a join permission

capability is able to confirm or deny a join request to an in-progress call by any potential participant. This capability should only be granted to one member within the whole communication. If more than one member is granted this capability, then extra mechanisms are essential to manage the potential conflicts.

The invite capability allows its holder to invite a potential member to join a call, while the exclude capability allows its member to exclude a participant from a call in progress. The control messages capability allows its holder to receive other miscellaneous control messages [Recommendation X.6].

2.7 Data Transfer

The starting of the data transfer phase depends on the identity of the call initiator and the multicast sender. If the call initiator is the sender, then the data transfer phase is entered as soon as the call is sent by the multicast service. If the call initiator is different from the sender, then the data transfer phase is only entered when the sender joins the multicast call. If the multicast call is established via static administrative means, then the data transfer phase is entered when the data sender interface is up [Recommendation X.6]. Figure 10 shows the starting point of the data transfer phase in a multicast call.

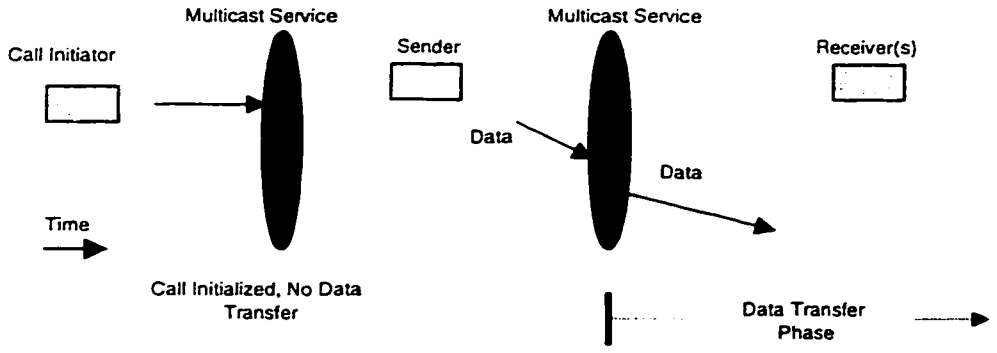

Figure 10: Data Transfer Phase

2.8 Data Integrity

Ideally, all the receivers of a multicast call should receive data units sent by any sender without corruption. However, data corruption or complete loss is possible due to many factors such as network

problems or buffer overflows. The data loss or corruption may occur between the sender and the multicast server or between the server and the receivers. Depending on the multicast session requirements, recovery against lost or corrupted data may be needed or not. If data recovery is needed, a loss or corruption between the sender and the multicast server would affect all the receivers and hence a recovery is to be applied to all the receivers. A loss or corruption between the multicast server and a receiver should be handled locally between these two entities. The details of data recovery methods are discussed in chapter 3, Reliable multicasting.

2.9 Multicasting at the Different Layers

2.9.1 Hardware Multicasting

Hardware Multicasting is a feature of the data-link hardware used to transmit packets to multiple hardware interfaces. Not every network hardware supports data-link Multicasting. Ethernet is one type of network hardware that supports data-link Multicasting.

2.9.1.1 Ethernet Multicast Addresses

Not all Ethernet frames are multicast datagrams. Ethernet frames have to be examined at the higher levels to detect whether they are unicast or multicast frames. As shown in figure 11, Ethernet unicast and multicast addresses are differentiated as follows: if the low-order bit of the high-order byte of the Ethernet address is a 1, then the address is a multicast address; otherwise it is a unicast address.
Ethernet multicast addresses are assigned dynamically by network protocols, while unicast addresses are assigned by the interface's manufacturer.

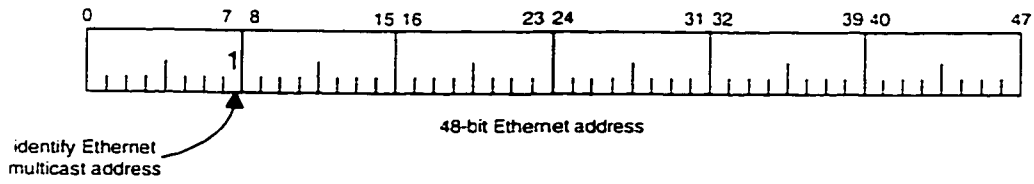

Figure 11: Testing for an Ethernet Multicast Address

If multicast transmission is needed, the higher-level protocol sets the unicast/multicast bit to 1, places an Ethernet group address as the destination address and transmits it on the wire. Hosts that are not interested in receiving multicast packets need only to ignore them. A hardware filter is used, at each host that is interested in receiving multicasts, to discard any unwanted datagrams, hence passing only the ones intended for that host. The hardware filter is usually built as a part of the Ethernet card. Without this hardware filter, each host would have to accept all multicast packets, examine them and then discard the unwanted ones.

2.9.2 IP Multicasting

IP Multicasting is a software feature implemented in IP systems to transmit packets to multiple IP addresses that may be located throughout the internet [Wright/Stevens 1995]. There are five classes of IP addresses. IP Multicasting is supported by class D addresses. Class A, B and C support unicasting, while the last class - class E – is still experimental. Table 1 shows the different classes of IP addresses.

Class	Range	type
A B C	0.0.0.0. → 127.255.255.255 128.0.0.0. → 191.255.255.255 192.0.0.0. → 223.255.255.255	Unicast
D	224.0.0.0. → 239.255.255.255	Multicast
E	240.0.0.0. → 247.255.255.255	Experimental

Table 1: Classes of IP Addresses.

Class D addresses do not identify individual interfaces in an internet. Rather, they identify group of interfaces, and so they are called multicast groups. A datagram with a class D address as its destination is delivered to all interfaces that have joined this corresponding multicast group.

Not all class D multicast groups are available for applications. For example, the first 256 groups (224.0.0.0 to 224.0.0.255) are reserved for protocols that implement IP unicast and multicast routing mechanisms. For example, group 224.0.0.1, known as all-hosts group, is the group for all systems on the subnet. Group 224.0.0.2, is the group for all routers on the subnet. Unicast and multicast routers may join this group to communicate with each other. Datagrams sent to these groups are not forwarded beyond the local network by multicast routers.

2.9.3 IP to Ethernet Multicast Address Mapping

An efficient implementation of IP multicasting would take advantage of the hardware multicasting; otherwise, as mentioned in section 2.9.1.1, each IP datagram is to be broadcast to the network and every host has to examine the received datagrams and discard the unwanted ones. However, for the hardware filter to work, IP multicast group destinations have to be converted by the network interface to link-layer multicast addresses that the hardware can recognize. On point-to-point networks, there is only one possible destination, so the mapping can be done implicitly. On other networks such as Ethernet, an explicit mapping method is required. As Ethernet supports multiple protocols, this method must be able to allocate the multicast addresses, yet prevent any conflicts. To support IP multicasting, a block of Ethernet addresses is assigned to the IANA (Internet Assigned Numbers Authority) [Wright/Stevens 1995]. All the addresses in that block start with 01:00:5e. Figure 12 illustrates how Ethernet multicast addresses are constructed from a class D IP addresses. The high-order 9 bits of the class D IP address are not used when constructing the Ethernet address.

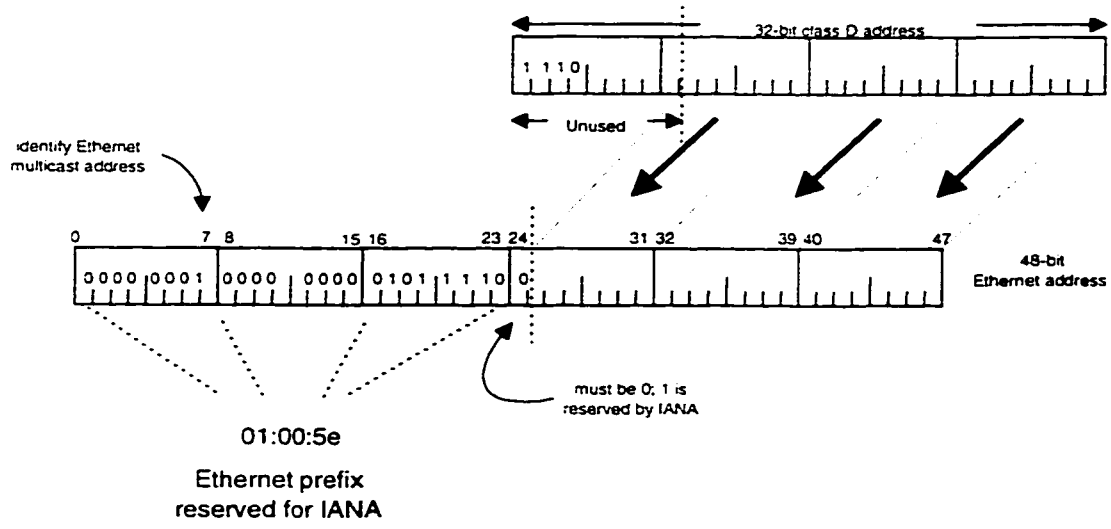

Figure 12: Mapping between IP and Ethernet Addresses

2.10 Group Membership Management

Group membership is dynamically determined as different network interfaces join or leave groups based on requests from processes running on each system. On a single network, the Internet Group Management Protocol (IGMP) conveys group membership information between systems. Group membership information is propagated by the multicast routers using some multicast routing protocol such as the Distance Vector Multicast Routing Protocol (DVMRP). A standard IP router may support multicast routing, or a dedicated router may be provided to handle the multicast routing.

2.10.1 IGMP: Internet Group Management Protocol

IGMP is used by IP systems (hosts and routers) on a local network to report their IP multicast group memberships to any neighboring multicast routers. Routers multicast IGMP queries periodically to the all-hosts group. Hosts respond to these queries by multicasting IGMP report messages. An IP multicast router may itself be a member of one or more multicast groups. In this case the router is capable of behaving both as a router – collecting membership information needed by the multicast protocol – as well as a typical group member – informing itself and other routers of its membership. From the architectural

perspective. IGMP is considered to be a transport-layer protocol that operates above IP. It has a protocol number (2), and its messages are carried in IP datagrams. Usually, IGMP is not accessed directly by a process, but processes can send and receive IGMP messages through an IGMP socket. This feature was provided to enable multicast routing daemons to be implemented as user-level processes [Wright/Stevens 1995]. Version 1, specified in RFC-1112, was the first version to become an Internet standard. Version 2, specified in RFC-2236, adds support for "low leave latency"; that is, a reduction in the time it takes for a multicast router to learn that none of the members of a specific group is still present on an attached network [Cain/Deering, 1999]. The latest version of IGMP, version 3, adds support for "source filtering". With this feature, a system is able to report interest in receiving packets sent to a particular multicast address, only from a specific source address or from all sources but specific ones.

IGMP functionality is not restricted to group management. The protocol is also used for other IP multicast management functions using other message types different from the ones used for group membership.

For group membership management, multicast routers use IGMP to learn which groups have members on each of their attached physical networks as follows: Initially, with respect to each of its attached networks, a multicast router may assume one of two roles: Querier or Non-Querier [Fenner, 1997]. A Querier router is capable of sending general queries on an attached network to solicit membership information. There is normally only one Querier per physical network. By default, each multicast router starts up as a Querier on each of its attached networks. Once started up, if a multicast router hears a query message from a router with a lower IP address, it changes its status and becomes a Non-Querier on that network. If a router has not heard a query message from another router for a specific time interval, it resumes the role of Querier. A router periodically sends general queries on each of its attached networks for which this router is the Querier, to solicit membership information. In order to quickly and reliably determine the membership information in the beginning, a Querier router sends many general queries spaced closely to each other; that is, they are separated by a small time interval [Fenner, 1997]. These general queries are addressed to the all-systems multicast group (224.0.0.1). When a host receives a query, it sets delay timers for each group,

excluding the all-systems group, that is a member on the interface from which the query was received. Each timer is set to a different random value between 0 and a maximum value. The maximum value is equal to a maximum response time value specified in the query. When a host receives a query for a specific group, it sets a delay timer to a random value selected as above for the group being queried if it is a member on the interface from which it received the query. If a timer for the group is already running, it is reset to the random value only if the maximum response time requested in the packet is less than the remaining value of the running timer. When a group's timer expires, the host multicasts a membership report to the group, with IP Time-To-Live (TTL) of 1. If the host receives another host's report while it has a timer running, it stops its timer for the specified group and does not send a report, in order to suppress duplicate reports [Fenner, 1997].

When a router receives a report, it adds the group being reported to the list of multicast group membership on the network on which it received the report. The router then resets a specific timer that is used to detect groups with no members. Repeated reports refresh this timer. If no reports are received for a particular group before this timer has expired, the router assumes that the group has no local members and that it need not forward remotely-originated multicast packets for that group onto the attached network [Fenner, 1997].

When a host joins a multicast group, it should immediately transmit an unsolicited membership report for that group, in case it is the first member of that group on the network. The member may transmit one or two duplicates of the report after a short delay to cover the possibility of the initial report being lost or damaged [Fenner, 1997].

When a host leaves a multicast group, if it was the last host to reply to a query with a membership report for that group, it sends a message indicating that it is leaving the group to the all-routers multicast group (224.0.0.2). If the host was not the last one to reply to a query, it is not required to send any messages, since there must be another member on the subnet. This is an optimization to reduce traffic. However, if the host

does not have sufficient storage to record whether or not it was the last host to reply, it should always send a message indicating that it is leaving the group [Fenner, 1997].

When a Querier receives a message indicating that a member is leaving a group and that group has group members on the reception interface, the Querier sends a message to the group indicating that it is about to consider that this group has no local members. The Querier then has to wait for a specific time interval. If no reports are received within this time interval, the router assumes that the member left this group was the last one, and hence the group has no more local members [Fenner, 1997].

2.11 IP Multicast Routing

IP Multicast is capable of transmitting packets from a particular source to a group of receivers via a spanning tree that connects all the hosts in the group. The techniques used to construct this spanning tree differ from one routing protocol to another. However, once the tree is constructed, all multicast traffic between the sender and the receivers is distributed over it.

IP Multicast routing protocols generally follow one of two basic approaches depending on the expected distribution of multicast group members throughout the network. The first approach is based on the assumptions that the multicast group members are densely distributed throughout the network and that bandwidth is plentiful. Dense distribution of members, referred to as "dense-mode", means that many of the subnets contain at least one group member. The second approach to multicast routing basically assumes that the multicast group members are sparsely distributed throughout the network and bandwidth is not necessarily widely available. This mode of communication is referred to as "sparse-mode". Sparse-mode does not imply that the group has a few members. It simply implies that the members are widely dispersed.

Dense-mode multicast routing protocols rely on a technique called flooding to propagate information to all network routers. Example of dense-mode routing protocols include DVMRP, the Multicast Open Shortest Path First (MOSPF), and Protocol-Independent Multicast - Dense Mode (PIM-DM).

With spare-mode multicast routing protocols flooding would unnecessarily waste network bandwidth and hence could cause serious performance problems. Hence, sparse-mode multicast routing protocols rely on more selective techniques to set up and maintain multicast trees. Sparse-mode routing protocols include Core Based Trees (CBT), and Protocol-Independent Multicast - Sparse Mode (PIM-SM). Sections 2.11.1 and 2.11.2 discuss briefly some of these IP routing protocols.

2.11.1 Dense-Mode Multicast Routing Protocols

2.11.1.1 DVMRP: Distance Vector Multicast Routing Protocol

DVMRP is a routing protocol that is used for routing multicast datagrams through an internet. The protocol is derived from the Routing Information Protocol (RIP) and implements multicasting as described in RFC-1054 [Waitzman/Partridge, 1988].

In a nationwide network such as the current internet, it is very unlikely that a single routing protocol will be used for the whole network. Rather, the network will be organized as a collection of autonomous systems. An autonomous system is most likely administrated by a single entity. Each autonomous system has its own routing technology, which can be different from one autonomous system to another as well. The routing protocol used within an autonomous system is referred to as "interior gateway protocol" (IGP). A separate protocol is used to interface among the autonomous systems [Hedrick, 1988]. DVMRP is an interior gateway protocol that is suitable for use within autonomous system, but not between different autonomous systems. The protocol is developed precisely for routing only multicast datagrams. Routers that support both multicast and unicast datagrams need to run two separate routing processes.

In fact, DVMRP uses IGMP to exchange routing datagrams. A DVMRP datagram is composed of two parts: a fixed-length IGMP header, and a stream of tagged data. Figure 13 shows a typical example of a

DVMRP datagram. The version field indicates the used protocol version while the type field indicates the DVMRP protocol type, which is 3. The subtype is 8-bit field, which can be 1, 2, 3 or 4 where:

1 = Response; the message provides routes to some destination(s).

2 = Request; the message requests routes to some destination(s).

3 = Non-membership report; the message provides non-membership report(s).

4 = Non-membership cancellation; the message cancels previous non-membership report(s).

The checksum is a 16-bit field and contains the one's complement of the one's complement sum of the entire message, excluding the IP header.

The rest of the DVMRP message is a stream of tagged data. The use of tagged data allows easy extensibility, since new commands can be developed by adding new tags, and reduces the amount of redundant data in a message. The elements in the stream, referred to as commands, are always multiples of 16 bits for convenient alignment. A command is organized as an 8-bit numeric code, with at least an 8-bit data portion. All commands have to align to 16 bits or a multiple of 16 bits as mentioned above.

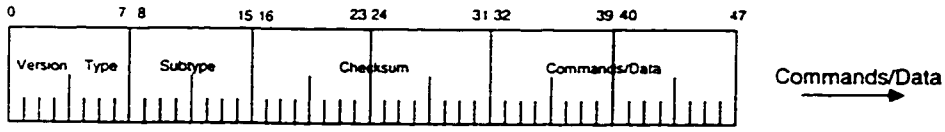

Figure 13: A Typical DVMRP Datagram

2.11.1.1.1 Constructing the Spanning Tree in DVMRP

DVMRP constructs a different distribution tree for each source and its destination host group. Each distribution tree is a minimum spanning tree from the multicast source, as the root of the tree, to all the multicast receivers, as leaves of the tree. The distribution tree provides a shortest path between the source and each multicast receiver in the group based on the number of hops in the path, which is the DVMRP metric.

Initially, DVMRP assumes that every host on the network is part of the multicast group. The designated router on the source subnet, the router that has been selected to handle routing for all hosts on its subnet, begins by transmitting a multicast message to all adjacent routers. Each of these routers then selectively forwards the message to downstream routers until the message is eventually passed to all multicast group members. To ensure that the tree will have no loops and that it will include the shortest paths from the source to all recipients, a mechanism called Reverse Path Forwarding is used. When a router receives a multicast message, it checks its unicast routing tables to determine the interface that provides the shortest path back to the source. If that was the interface over which the unicast message arrived, then the router enters some state information to identify the multicast group in its internal tables and forwards the multicast message to all adjacent routers, other than the one that sent the message. Otherwise, the message is simply discarded.

Finally, the protocol eliminates branches of the tree that don't lead to any multicast group members. The IGMP, which is running between hosts and their immediately neighboring multicast routers, is used to maintain group-membership data in the routers. When a router determines that no hosts beyond it belong to the multicast group, it sends a message to its upstream router indicating that. As a result, routers update the state information in their tables to reflect which branches have been removed from the tree. This process continues until all superfluous branches are eliminated from the tree, resulting in a minimum spanning tree.

Figure 14 illustrates the construction of a DVMRP spanning tree. Hop/time units measure the progression of messages as follows:

1. In the first hop the message reaches router 1.

2. In the second hop the message reaches routers 2,3, and 4.

3. In the third hop packets reach routers 5, 6 & 8. In addition, routers 3 and 4 exchange messages. Each one just drops the message, because it didn't arrive over the interface that gives the shortest path back to the source.

4. In the fourth hop the message reaches router 7. Router 7 realizes it is a leaf router and there are no group members on its subnet, so it sends a remove message back to router 6, the upstream router. Router 6, in turn, sends a remove message to router 4. Router 3 also sends a remove message to router 1.

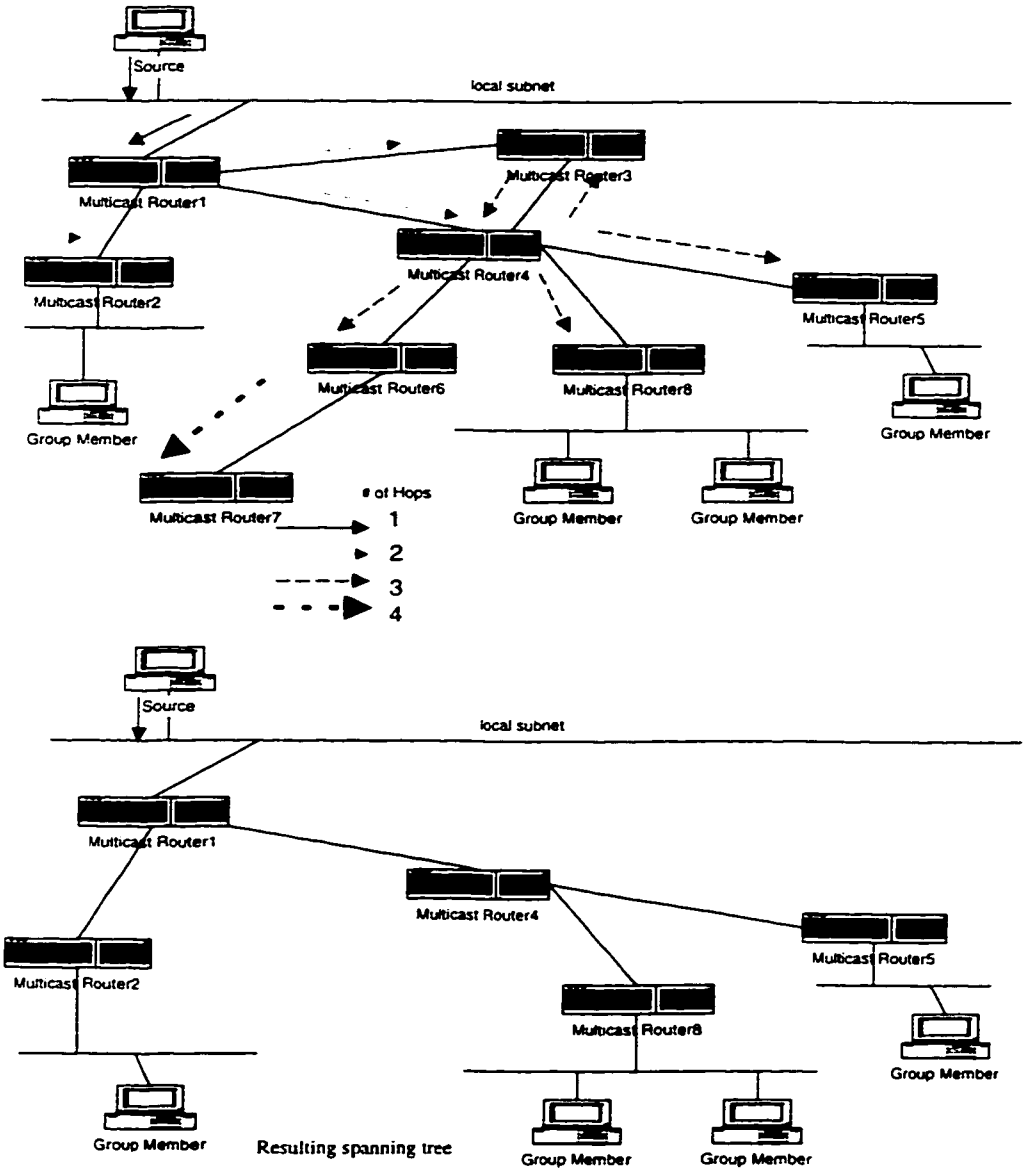

Figure 14: Constructing a DVMRP Spanning Tree.

2.11.1.2 PIM-DM: Protocol-Independent Multicast - Dense Mode

PIM-DM is similar to DVMRP. It also employs Reverse Path Multicasting to construct source-rooted distribution trees. The major difference between PIM-DM and DVMRP is that PIM-DM is completely independent of the unicast routing protocol that is used on the network, while DVMRP relies on specific mechanisms of the associated unicast routing protocol. In general, PIM-DM is less complex than DVMRP [Johnson/Johnson, 1996].

Similar to all dense-mode routing protocols, the PIM-DM protocol is data driven. However, since PIM-DM is independent of the accompanying unicast routing protocol, data packets which arrive at a router over the proper receiving interface, the interface that provides the shortest path back to the source, are forwarded on all downstream interfaces until unnecessary branches of the tree are explicitly removed. The philosophy followed by PIM-DM designers is to opt for protocol simplicity and protocol independence, even though there is likely some additional overhead due to some packet duplication [Johnson/Johnson, 1996].

2.11.2 Sparse-Mode Multicast Routing Protocols

Sparse-mode multicast routing protocols can not rely on the periodic flooding of messages throughout the network like dense-mode routing protocols. The flooding approach can be considered efficient within regions where a multicast group is widely represented throughout the network or where there is lots of available bandwidth. However, for cases such as when several thousand small conferences are being conducted simultaneously over the Internet, the aggregate traffic from this periodic flooding can potentially saturate wide-area Internet connections. Hence, sparse-mode protocols must rely on different approaches that restrict the multicast traffic to those links that lead to members of the multicast group.

To construct multicast distribution trees, sparse-mode protocols use a receiver-initiated process. That is, a router becomes involved in the construction of a multicast distribution tree only when one of the hosts on its subnet requests membership in a particular multicast group.

2.11.2.1 CBT: Core Based Trees

Unlike DVMRP, which constructs a shortest-path tree from each source to all the group destination members, the CBT protocol constructs a single tree that is shared by all members of the group. Multicast traffic for the entire group is sent and received over the same tree, regardless of the source. This use of a shared tree can provide significant savings in terms of the amount of multicast state information that is stored in individual routers [Johnson/Johnson, 1996].

A CBT shared tree has a core router that is used to construct the tree. Routers join the tree by sending a join message to the core. When the core receives a join request, it returns an acknowledgment over the reverse path, thus forming a branch of the tree. To eliminate unnecessary traffic, join messages need not travel all the way to the core before being acknowledged. If a join message hits a router that is already on the tree before it reaches the core, that on-tree router acknowledges the request and terminates it without forwarding it any further. The acknowledgement of the request allows the requester router to be connected to the tree. Figure 15 shows an example of a CBT tree and illustrates the join/acknowledge operation.

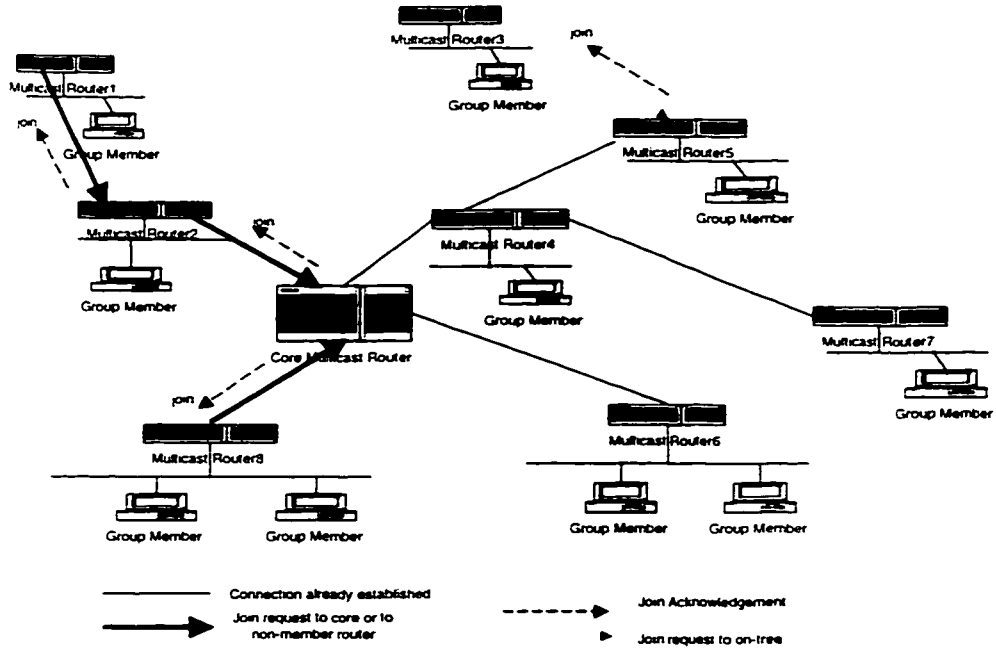

Figure 15: Joining a CBT Shared Tree

The biggest potential problem of CBT is the traffic concentration around the core. Some versions of CBT dealt with this problem by supporting multiple cores, and implementing some techniques for load balancing among these cores.

2.11.2.2 PIM-SM: Protocol-Independent Multicast - Sparse Mode

PIM-SM was designed to provide an efficient routing to multicast groups that may span wide-area, and inter-domain, internets [Wei/Estrin, 1999]. PIM-SM constructs a distribution tree for each group. A router receives explicit Join/Prune messages from those neighboring routers that have downstream group members. The router then forwards data packets addressed to a multicast group only onto those interfaces on which explicit Joins have been received. A Designated Router (DR) sends periodic Join/Prune messages toward a group-specific Rendezvous Point (RP) for each group for which it has active members [Wei/Estrin, 1999]. The Rendezvous Point is the point where sources and receivers learn of each other.

Each router along the RP builds a wildcard (any-source) state for the group and send Join/Prune messages on toward the RP [Wei/Estrin, 1999].

PIM-SM utilizes IGMP to convey membership messages. Should a host wish to join a multicast tree, it conveys its membership information through IGMP [Wei/Estrin, 1999]. When there are no longer directly connected members for the group, IGMP notifies the DR.

PIM-SM allows routers to switch from a shared tree (RP-tree) to a shortest path tree (a SP-tree). A router with a directly-connected member first joins the shared RP-tree. The router can switch to a SP-tree after receiving packets from that source over the shared SP-tree [Wei/Estrin, 1999]. The recommended policy is to initiate the switch to the SP-tree after receiving a significant number of data packets during a specified time interval from a particular source [Wei/Estrin, 1999].

2.11.3 Tunneling Strategy for IP Routing

Unfortunately, not all parts of an internet may be multicast-capable. Many routers within an internet may be capable only of supporting unicast packets. Without the tunneling strategy, these non-multicast capable parts would be a cap for IP routing. With tunneling, multicast packets are encapsulated in an IP datagram, such as a unicast packet, before they are routed through any non-multicast capable part of the network. That is, unicast encapsulated multicast packets are forwarded through these non-capable multicast parts as if they were just regular unicast datagrams. The encapsulation is added on entry to a tunnel and stripped off on exit from a tunnel. The most well-known demonstration of multicast tunneling is used in the Multicast Backbone (MBONE), with DVMRP [Johnson/Johnson, 1996].

2.11.4 The Multicast Backbone MBONE

As mentioned in the previous section, not all routers on the internet support IP multicast routing. The set of the Internet routers, which supports IP multicast routing, composes the MBONE. The MBONE exists to support experimentation with IP multicast – in particular with audio and video data streams

[Wright/Stevens, 1995]. In fact, The MBONE is an outgrowth of the first two Internet Engineering Task Force (IETF) "audiocast" experiments in which live audio and video were multicast from the IETF meeting site to destinations around the world [Casner, 1993]. The MBONE can be views as a virtual network that is layered on top of portions of the physical Internet to support routing of IP multicast packets. The network is composed of many sub-networks that directly support IP multicast. The sub-networks are linked together via "tunnels" as explained in section 2.11.3. A tunnel is a virtual point-to-point link which connects parts of the MBONE. The tunnel endpoints are typically workstation-class machines having operating system support for IP multicast and running the multicast routing daemon "mrouted".

In the MBONE, threshold values limit how far various data streams propagate. This operation is controlled via setting a Time-To-Live (TTL) value for the IP multicast datagrams. Each category of application has a recommended TTL value. Table 2 shows the early recommendation of the TTL values for various applications along with the scope in which this TTL value should be associated. It should be noticed that the information provided by the table might not be very accurate today.

TTL	Application	Scope
0		same interface
1		same subnet
31	local event video	
32		same site
63	local event audio	
64		same region
95	IETF channel 2 video	
127	IETF channel 1 video	
128		
159	IETF channel 2 audio	same continent
191	IETF channel 1 audio	
223	IETF channel 2 low-rate audio	
255	IETF channel 1 low-rate audio	
	Unrestricted in scope	

Table 2: TTL Values for IP Multicast Datagram

As shown in the table, the TTL controls how far the data stream can propagate. For example, to restrict local event video datagrams from propagating outside of the local net, the TTL for these datagrams is set to 32 or less, while the interface that communicates to the outside network is configured with a multicast threshold of 32. As there is at least one hop between the source and the router, when the datagram reaches the interface its TTL value is less than 32, and so it is discarded. A multicast datagram that starts with a TTL of 128 and able to reach the interface within 96 (128-32) hops, would pass through site interfaces with a threshold of 32, but would be discarded by international interfaces with a threshold of 128 for example.

2.12 IP Multicasting over ATM

The increasing popularity of the Asynchronous Transfer Mode (ATM) in recent years was the force to much cooperative work to allow compatible and interoperable implementations for transmitting IP datagrams and ATM Address Resolution Protocol (ATMARP) requests and replies over ATM Adaptation Layer 5 (AAL5). The first recommendation, RFC-1577, was published early 1994. However, the recommendation was concerned only with classical IP and ARP over ATM. It is quite possible that multicasting has been intentionally left out of the scope of the recommendation because of the difficulty of translating IP's abstract group addresses to ATM point-to-multipoint virtual channel connections (VCCs). The IP approach to multicasting is very different from the one used in ATM. In IP on a broadcast medium, such as Ethernet, the joining/leaving of a host group is a local operation - the joining host simply starts/stops listening to the multicast transmission. Since ATM uses connection-oriented approaches, multicasting is done using a point-to-multipoint connection, which must be explicitly set up before it can be used for the transmission of data. The connection can only be initiated and modified by a single node, which functions as the root node of the multicast tree [Oosthoek, 1997].

Surely, the above mentioned difficulties could not terminate the effort for further research to allow IP multicasting over ATM. In 1996, the IETF RFC-2022 was the first attempt to provide IP multicasting directly on top of ATM - in particular over UNI 3.0/3.1 ATM based networks. The recommendation introduced new aspects and concepts to achieve ATM level multicasting of layer 3 packets. Examples of

these aspects include the Multicast Address Resolution Server (Mars), the multicast Cluster, and the Virtual Channel (VC) Mesh and the Multicast server (MCS) approaches. The following sections discuss the major concepts introduced in the recommendation.

2.12.1 The Multicast Address Resolution Server (MARS)

As the name may indicate, the Multicast Address Resolution Server (MARS) is the entity responsible for the resolution of IP multicast Class D addresses. Additionally, MARS also keeps track of group membership information. In fact, MARS is an extended analog of the ATM ARP Server introduced in RFC-1577 [Laubach, 1994]. It acts as a registry that associates layer 3 multicast group identifiers with the ATM interfaces representing the group's members. MARS messages support the distribution of multicast group membership information between MARS and the endpoints (hosts or routers). Endpoint address resolution entities query MARS when a layer 3 address needs to be resolved to the set of ATM endpoints that make up the group at any one time. Endpoints keep MARS informed when they need to join or leave any particular layer 3 groups. To provide asynchronous notification of group membership changes, MARS manages a point-to-multipoint VC out to all the endpoints that desire multicasting support [Armitage, 1996].

2.12.2 The ATM Level Multicast Cluster.

An ATM multicast Cluster is defined as the set of ATM interfaces choosing to participate in direct ATM connections to achieve multicasting between themselves. In practice, a Cluster is the set of endpoints that choose to use the same MARS to register their memberships and receive their updates [Armitage, 1996].

The communication between interfaces belonging to different Clusters is performed via inter-cluster devices. An inter-cluster device is analogous to a router in the context of IP.

2.12.3 Intra-cluster Multicasting

Two different approaches exist to achieve intra-cluster multicasting. The first approach, called VC-mesh, achieves intra-cluster multicasting through meshes of point-to-multipoint VCs. The second approach, MCS, achieves intra-cluster multicasting via the utilization of a separate ATM level multicast server.

2.12.3.1 VC meshes (VC-mesh)

With VC-mesh, each source establishes its own independent point-to-multipoint VC to each node in the destination group, which it should multicast data to. This constructs a single multicast tree from the sender to all the receivers in that designated group. Interfaces that are both senders and group members to a given group originate one point-to-multipoint VC, and terminate one VC for every other active sender in the group [Armitage, 1996]. Figure 16 shows an example VC-mesh structure.

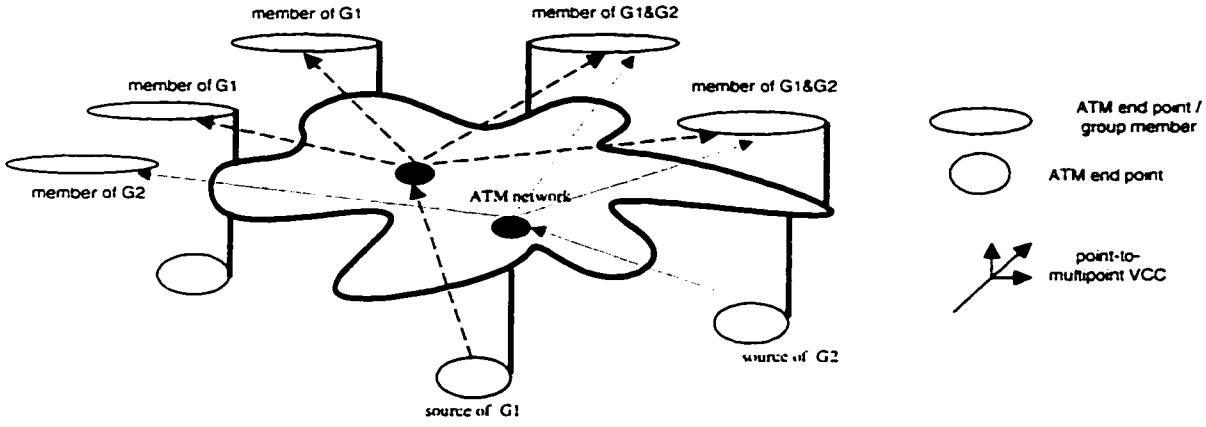

Figure 16: An Example of a VC-mesh Structure

The VC-mesh structure has a few advantages. The structure has no single point of failure/congestion. The probabilities of packet loss/damage between senders and receivers is low as there is a direct point-to-multipoint VCC from a sender to the host group members. That is, a sending host is not a leaf on the point-to-multipoint connection that it originates, and hence there are no reflected packets. In addition, there is no

problem of having multiple senders sending at the same time, because each sender has its own FIFO point-to-multipoint VCC to all leaves.

Unfortunately, these advantages of VC-mesh do not come for free. In general, the approach makes a heavy use of connection table entries at the end-stations and in the ATM switches. Any change to the group requires all existing members to initiate *add* and *remove* messages to reflect the changes. If the group is large, this process can be very time consuming and it may result in an overload of the protocols in the control plane of the network. In addition to these disadvantages, the VC-mesh approach may be costly to use if the ATM provider charges per VC, as a result of the high number of connections through the ATM switch. Point-to-multipoint VCCs require more resources in the switch.

2.12.3.2 Multicast Server (MCS)

The MCS approach provides an alternative to the VC-mesh for achieving multicasting. The major difference between the two models is in the way the connections are formed between the sender and the destination group members. With MCS, each source establishes a VC to an intermediate node - the multicast server (MCS), rather than to each member in the destination group. The multicast server then establishes and manages a point-to-multipoint VC out to the actual desired destinations. The MCS acts as a proxy server which multicasts data received from a source to the group members in the Cluster. Thus, each multicast source maintains only a single point-to-multipoint VC to the designated MCS for the group. The designated MCS terminates one point-to-multipoint VC from each cluster member that is multicasting to the layer 3 group. Each group member is the leaf of the point-to-multipoint VC originating from the MCS [Talpade/Ammar, 1997]. Figure 17 shows an example of a MCS structure.

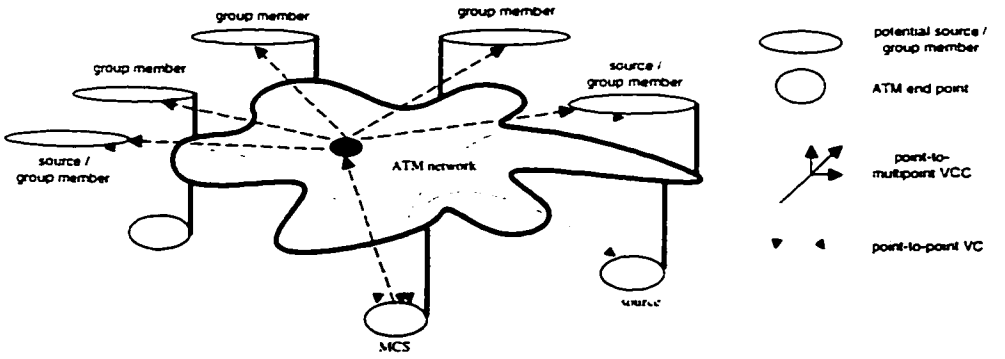

Figure 17: An Example of a MCS Structure

The advantages of MCS can be summarized as follows. The needed number of connection table entries for each host is not more than two. (It is two when the host is a group member as well as a sender to that group.) Changing the host group memberships is very easy since only the MCS needs to be informed when adding to or removing from the multicast tree occurs. The resource consumption in the switches at the client-side is low; however the switch connected to the MCS needs to process all temporary connections to and from the MCS.

On the other hand, the obvious disadvantage of MCS model is the concentration around the MCS server itself. The MCS forms a bottleneck and a single point of failure in the multicast transmission. Having multiple active MCSs along with some backup MCSs that can be used if failure occurs with any active MCS can reduce these problems. However, the MCS model has other disadvantages. Depending on the MCS location, the packet routing can be very inefficient. It is very possible that packets may have to traverse a long way to and from the MCS before it reaches the destination hosts. It is also difficult to handle simultaneous multiple sends, because standard AAL5 has no packet ordering method. The receiving ends expect all parts of a single packet to come in sequence and back to back on the VC. This can cause delays in the MCS when multiple hosts are sending at the same time [Oosthoek, 1997]. It also requires the MCS to be equipped with large buffers to avoid the loss of any packets. A final disadvantage of MCS is that it forwards the packets to all members, which may include the sender. The sending host then has to filter out its own transmissions.

Unfortunately, it is not so clear which of the two models - VC-mesh and MCS - is preferable. The choice is very dependent on both the situation and the QoS required. For example, in circumstances where the group is small and stable, it might be better to use a VC-mesh. However, if the group is relatively large or it changes often, then MCS might be a better choice. In general, which approach to use is a decision that has to be made by the system administrator after examining the trade-offs between latency, throughput, congestion and resource consumption.

Chapter 3

Multicasting – The Reliability Question

For some applications, such as most video and audio applications, reliable data delivery may not be a big concern. The data in these applications is transient, so that skipping over missing segments can be a sufficient recovery against lost/damaged packets. However, for many other collaborative applications, such as Jacobson's shared whiteboard tool (wb), shared text editors, shared databases, detailed image analysis and image archiving, the data sets are considered to be more static and some sort of reliable delivery is necessary to insure consistency across multiple views. For these applications, the "best effort" delivery provided by IP multicast is precisely interpreted as "insufficient effort". As a result, other multicast protocols needed to be developed to provide that required reliability. Yet, the design of each of these protocols would be impossible without answering first one major question: "What is reliability?". Still, another important question is present: "In which layer(s) of the protocol stack should reliability functions be implemented?". The answers to these questions are the subject of the next two sections. The rest of the chapter examines many other important issues related to reliable multicasting.

3.1 What is Reliability?

Garcia and Spauster formed the first answer to this question in 1991. From their perspective, reliability is determined by three properties of a protocol: delivery time, delivery atomicity, and ordering. Delivery time concerns with the length of time it takes to deliver a multicast message. This time can be measured either from message initiation at the source or from the moment the first message arrives at a group member. Delivery time may be guaranteed at one group member, a subset of the group members, or all group members. The second property, atomicity, requires that all group members deliver a message to the application within a specified time interval. This time interval can start either when one group member has just delivered the message or after a majority of the members have delivered that same message. The final property, ordering, concerns the order in which the packets are delivered. Different levels of ordering

exist, starting from no guarantees of ordering, to a full guarantee of ordering, which means that messages are always delivered consistently even in worst circumstances such as the presence of network failures.

In 1992, Rajagopalan gave another definition of reliability. According to that definition, protocols are reliable if they offer completeness and finiteness. Completeness means that the protocol delivers multicast messages in the same order they are sent by the source, without message duplication or loss [Rajagopalan, 1992]. Unlike the definition of the ordering property given by Garcia and Spauster, the completeness definition did not mention anything concerning ordering guarantees in the presence of failures. Finiteness is similar to the atomicity property given by Garcia and Spauster.

In 1994, Bormann defined reliability from both the senders and receivers perspectives. From the sender's perspective, a protocol is reliable if the sender is assured, with sufficient probability, that all of its messages reach, within a bounded time, all recipients that are not failing or being partitioned [Bormann, 1994)]. From the receiver's perspective, a protocol is reliable if a receiver can determine when it is failing or being partitioned from active senders.

In 1996, Macker viewed reliability as a subject that is dependent on the application. With that vision, reliability is considered a common requirement for many data types in advanced distributed simulations and in situational awareness dissemination [Macker/Klinker, 1996]. In these types of applications, data may be in either an active or quiescent state. Reliability is not really needed while the object is in active state. This argument may hold because if the object is changing relatively quickly, lost or damaged data is simply refreshed with the next update. When a data object finally enters a steady state or becomes quiescent, its state is periodically advertised to the multicast group and reliability actions are performed only if an inconsistency is detected. However, this vision by Macker is not really considered by more conventional reliable protocols, where error recovery is initiated as soon as a lost or misordered packet is detected.

3.2 In Which Layer(s) of the Protocol Stack should Reliability Functions be Implemented?

How much reliability to provide at the different levels of the protocol stack should be decided by the application itself. Truly, many arguments hold for moving some of the reliability functions upward in a layered system towards the application that uses the functions. In practice, not all applications need the same amount of error checking at the lower levels. Many applications, such as financing and banking applications, do perform error checking themselves, hence do not need a perfect reliability service from the communication protocol. For these applications, reducing the delays introduced by error checking at the lower levels might be more beneficial. In addition, a complete and correct implementation of these functions may not be feasible without the knowledge and the help of the application. Hence, providing these functions completely as a part of the communication system itself may not be possible.

In general, all applications that demand end-to-end reliability require part of the implementation to be done at the higher levels. For example, a file transfer operation is performed by reading the file first from disk, passing it to the file transfer program, and then to the communication system for delivery. If the file is altered during the first two steps, then the delivered copy is surely different from the original, even if the communication system is perfectly reliable. In such situations, only end-to-end checks at a higher level than the communication system can guarantee reliability.

Another argument for moving the reliability functions higher in the protocol stack is known as Application Level Framing (ALF). ALF is discussed in the next section.

3.2.1 Application Level Framing (ALF)

ALF is a protocol architecture that has been proposed to allow efficient implementation of communications with diverse service requirements. ALF allows the application to control the way in which network errors are handled. The ALF concept argues for changing the design focus for protocols from the

transmission of units that are understood by lower layers in the protocol stack to units that are meaningful to the application. Protocols designed according to ALF principles break the data into Application Data Units (ADU's), which can be processed by the application whether they are in or out of order. The next subsections make clear how this argument may hold. Subsection 3.2.1.1 provides some background knowledge related to the subject, while subsection 3.2.1.2 provides a description and a conclusion of the ALF concept. Section 3.2.2 concludes an answer to the question in-hand.

3.2.1.1 Protocol Anticipation in Data Communication

The core function of protocols is to transfer application information among machines [Clark, 1990]. The functions performed during this data transfer can be generally categorized as either transfer control or data manipulation. Transfer control functions are concerned with operations related to the transfer of data. Examples of those functions include: flow/congestion control, detecting network transmission problems, acknowledgment sending and processing, multiplexing, and packet timestamping.

Data manipulation functions are those which either read or modify the data. Examples of those functions include: moving data to and from application address space, encryption, moving data to and from the network, detecting errors, buffering data for retransmission, and presentation formatting. Presentation formatting is defined as the reformatting of data into some common or external data representation [Clark, 1990].

The overhead of data manipulations is much more than that caused by transfer control. That is because data manipulations are much more processing intensive, since they involve touching all the bytes in the packet, perhaps several times [Clark, 1990]. In fact, a major part of the data manipulation overhead is caused by some operation called presentation conversion. Presentation conversion is a combination of presentation formatting and the operations of moving data to and from application address space.
A key aspect of presentation conversion is that it needs to be done in the context of the application [Clark, 1990]. For example, when a file is transferred, the received data is simply placed in sequential locations in

the application address space. However, for more complicated operations than file transfers, only the application can figure out what the sequence of data items is. Hence, the actual sequence of presentation conversions must be driven by application knowledge. This indispensable relationship between presentation conversion and the application makes the application a potential bottleneck in overall network throughput [Petitt, 1996]. Whenever data is lost or misordered, presentation conversion is interrupted since incoming packets are not delivered to higher layers in the protocol stack until they have been properly sequenced by the protocol. The amount of time needed to achieve proper sequence may be comparatively large since a whole retransmission by the sender may be required. Thus, a lost packet stops the application from performing presentation conversion, and to the extent when it is the bottleneck, the application can never catch up [Clark, 1990]. To avoid this problem, the protocols should be designed so that the application is not prevented from performing presentation processing as the data arrives [Clark, 1990].

To this end, any effort to enhance the protocol performance should primarily focus on presentation conversion. This is exactly the principle of ALF.

3.2.1.2 The Concept of ALF

As we previously mentioned, protocols designed according to ALF principles break the data into Application Data Units (ADU's), which can be processed by the application whether they are in or out-of-order. The reason why this argument holds is that presentation processing can continue uninterrupted in spite of out-of-sequence ADU's. The receiving application can adjust the out-of-order ADU's if it knows where to put the arriving data. This information can be communicated by the sender who must be able to specify the disposition of the ADU in terms meaningful to the receiver [Clark, 1990)]. For example, in the case of video transmission, the sender and receiver might agree on a code that identifies both the frame to which an ADU belongs and its location within that frame [Clark, 1990]. Similarly, in a file transfer, the sender can transmit both the ADU and the ADU's location within the receiver's file. In this case, the transport protocol does not have to be concerned with the packet ordering. Out of sequence ADU's can be

reordered by the application itself after delivery, since the application can determine their proper place within the receiver's file.

In 1996, Floyd and Jacobson expressed another support to the use of ADUs in multicast communication. A comparison between reliability in unicasting and multicasting indicated that the vocabulary used by the sender and the receiver in unicast communication was one of those conventions that migrated poorly to multicasting. In unicasting, a receiver can request a retransmission either in terms of a shared communication state such as a sequence number, or in terms of ADUs such as "sector *x* of file *y*". Although both models have been used successfully in unicasting, the shared communication state is by far more popular since it allows the protocol to be entirely independent of any application's namespace. However, with multicasting, the use of shared communication state does not work well, since multicast transmission tends to have much weaker and more diverse state synchronization than does unicast [Floyd/Jacobson, 1996].

For example, if a receiver joins a multicast conversation late and receives packets with sequence numbers 1000 to 1050, there will be no way for that receiver to find out what packets have been missed since the starting number from the sender is arbitrary. Hence, the receiver can not either do anything useful with the received data, or be able to request the proper missed packets. Thus, the use of the ADU's naming model works much better for multicasting.

To conclude, ALF have put another valid force to the shifting of some of the reliability functions up in the protocol stack towards the application itself. This wan indicated clearly by Floyd and Jacobson in 1996,who stated that ALF says that the best way to meet diverse application requirements is to leave as much functionality and flexibility as possible to the application.

3.2.2 The Question's Comeback

Although there is no clean cut answer to the question in-hand, it is clear that some of the reliability functions should be implemented at the transport layer while others are best handled at the higher layers in the protocol stack. In fact, implementing a transport layer that is able to handle all reliability aspects for every existing application is best expressed as "next to impossible".

3.3 Major Reliable Multicast Issues

3.3.1 Sender-Initiated Error Recovery

Sender-initiated reliable multicast protocols require the sender to have a full knowledge of the receivers. The sender then maintains a list, called the ACK list, for each transmitted packet. The ACK list is simply a list of the receivers from which ACKs have been received. Each time a receiver correctly receives a packet, it returns an ACK to the sender. Upon the receipt of the ACK, the sender updates the ACK list for the corresponding packet. Lost packets are detected via the use of timers. When the sender transmits a packet, it starts a timer. The ACKs should then be received within a specific configurable time. If the timer goes off before an ACK is received from any receiver, then the sender assumes that the packet to that receiver was lost, and hence it retransmits the packet to the receiver and starts the timer again.

In most traditional protocols, when the sender wishes to transmit/retransmit a packet, it multicasts it to all receivers. When a receiver receives a packet correctly, it returns the ACK to the sender over a point-to-point connection. The sender uses a selective repeat approach for retransmissions; only lost/corrupted packets are retransmitted.

The sender-initiated approach could be the most appropriate in situations where the sender must be in entire control or must have a full knowledge of the receivers. However, the approach has a major problem: implosion. As the number of receivers increases, the number of ACKs received at the sender is also

increased. After some threshold, the number of returned ACKs may exceed the ability of the sender to process them. Hence, the approach must be limited to small multicast groups, unless some techniques are adopted to reduce the number of ACKs returned to the sender. This latter issue will be explored in detail in the next chapter.

3.3.2 Receiver-Initiated Error Recovery

With receiver-initiated error recovery, the responsibility of ensuring reliable packet delivery is placed on the receivers. The sender just needs to continue transmitting new packets until it receives a negative acknowledgment (NACK) from a receiver. Each receiver is responsible for maintaining a reception state for itself. Upon the detection of an error, the receiver sends a NACK to the sender indicating the error. Typically, a receiver detects a packet loss via the use of sequence numbers. If the receiver receives a sequence number that is larger than expected, it presumes the occurrence of an error and hence sends a NACK. The sender then retransmits the requested packet to that receiver via the unicast connection between both of them, or possibly via multicasting the packet to a group of receivers that includes the requester.

In order to secure against the loss of the NACK, the receiver uses timers in a fashion similar to the one used by the sender in the sender-initiated approach. In special cases where the sender does not have packets to send (i.e., must occasionally wait for data to be produced by higher level), it may be necessary for the sender to multicast periodic state information, giving the sequence number of last transmitted packet for example [Towsley/Kurose, 1997].

In fact, there are a few variations of the receiver-initiated error recovery approach. Section 3.3.3 explores these variations and gives a comparison of sender-initiated and receiver-initiated error recovery approaches.

3.3.3 Variations of Receiver-Initiated Error Recovery

3.3.3.1 Sender-Oriented Receiver-Initiated Error Recovery

Two models of the sender-oriented approach exist. The distinction between the two models lies on the way NACKs are transmitted to the sender. With the first model, referred to as unicast NACK (N1), error detection at a receiver results in a NACK sent to the sender. Although intermediate receivers may have received the data for which the NACK was issued, only the sender is involved in issuing repairs [Macker/Klinker, 1996]. This model is appropriate when receivers have no way to communicate with each other, or when they are not allowed to do so for security reasons for example. However, this approach limits scalability because of the potential NACK implosion, which can occur if the receiver sets become large. Thus, the approach is best suited for transmission of very large packets where a low ratio of NACK overhead to data content can be realized [Macker/Klinker, 1996].

To explore the effects of unsuppressed NACK implosion, Macker and Klinker simulated a set of symmetric multicast trees with an independent probability of packet loss ρ due to congestions at tree nodes. Each multicast tree of depth d and fan-out n contains members at branch and leaf nodes. Upon a packet loss at a node, each child of this parent produces a NACK message. The total number of NACK messages is obtained and averaged over a large set of trials. 100,000 trials were performed for nodes with a fan-out of 4 (n=4), and a tree depths d=4,5, and 6. Figure 18 shows the result of the experiment. As expected, the NACK implosion effect is quite apparent as the size of the tree and the packet loss per node increase. Even in cases where raw link error rates are very low, the possibility of packet loss can be quite high, due to router congestion and other effects.

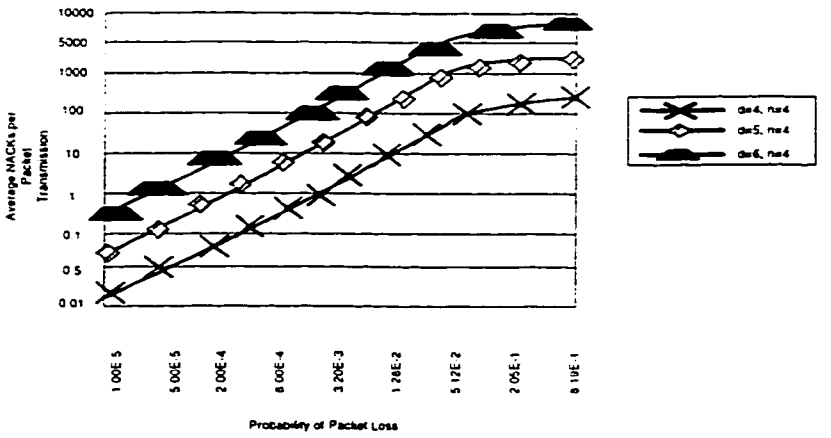

Figure 18: Simulation of NACK Implosion

Although the analysis performed by Macker and Klinker in 1996 proved that this error recovery approach might not scale to large receiver sets, an earlier study by Pingali in 1994 proved that this approach may still produce a better performance than the sender-initiated approach. The study used simulations to compare the maximum throughput of sender-initiated error recovery to unicast NACK sender-oriented/receiver-initiated approach at both the sender and the receivers. The throughput progress is given as a ratio between the throughput using receiver-initiated/sender-oriented protocol (N1) divided by the throughput using the sender-initiated protocol (A). That is, values greater than 1 indicate higher throughput. Figures 19 and 20 show the result of the simulation at the sender and at the receivers respectively. The figures also show the result of the simulation for different probability of packet loss. Loss probabilities of 1, 5, 10, 25, and 50 percent were simulated.

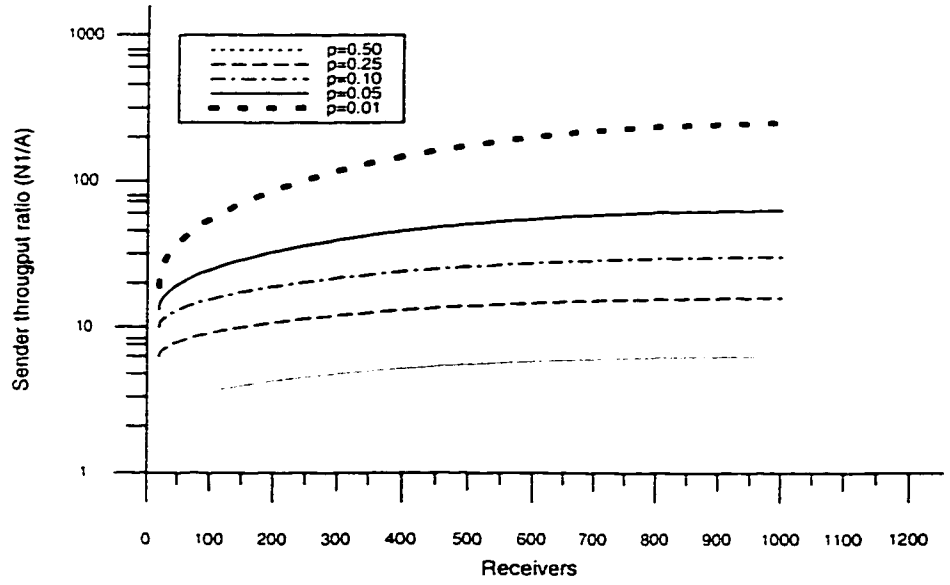

Figure 19: Sender Throughput Ratio (N1/A) vs Number of Receivers

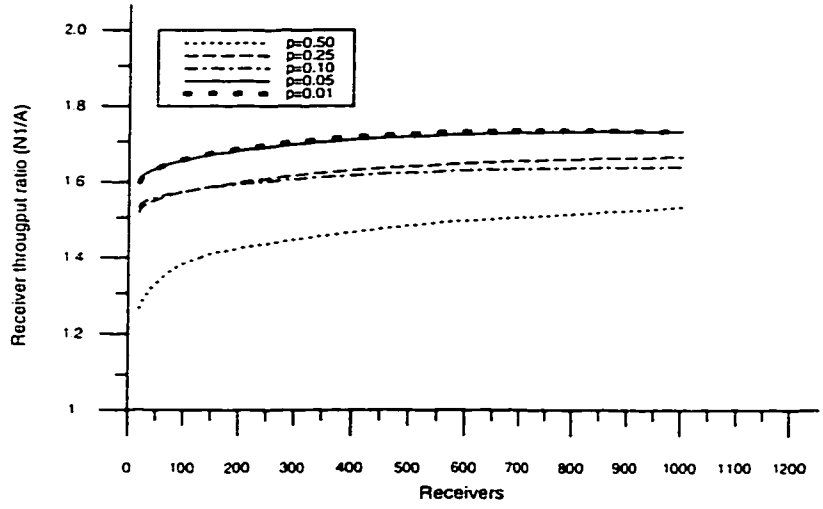

Figure 20: Receiver Throughput Ratio (N1/A) vs. Number of Receivers

As figure 19 shows, the receiver-initiated approach always improves the throughput at the sender over the sender-initiated approach. The reason behind this improvement is due to the amount of processing that each requires at the sender. Receiver-initiated/sender-oriented approach places a light burden on the sender

because processing is only required when losses occur. The sender-initiated approach, however, requires a continuous processing at the sender even if the communication is error free, as acknowledgments must be processed at the sender. This processing is also increased when the number of receivers is increased.

As shown in figure 20, the receiver throughput also improves with the receiver-initiated approach. This is due to the minimal processing requirement at the receiver for the receiver-initiated approach, as receivers only need to issue NACKs when losses occur. The sender-initiated approach requires receivers to acknowledge each packet.

The second model, referred to as broadcast NACK (N2), strives to reduce the number of NACKs received at the sender, and hence avoids the implosion problem. The model attempts to ensure that at most one NACK is returned to the sender by packet transmission. In that model, when a receiver detects an error, it broadcasts a NACK to the sender and all the receivers. However, the receiver has to delay itself for a random period of time before broadcasting the NACK. If within this delay time, the receiver receives a NACK that corresponds to the same packet it has missed, it simply cancels its NACK broadcasting. Repairs then are multicast to the group by the sender. This model is more scalable, hence it is more suitable than the unicast NACK when the receiver sets are large.

A similar simulation was performed by Pingali to compare the maximum throughput at the sender of sender-initiated approach to broadcast NACK sender-oriented/receiver-initiated approach, as well as of broadcast NACK sender-oriented/receiver-initiated to unicast NACK sender-oriented/receiver-initiated. Figures 21 and 22 show the result of these simulations respectively.

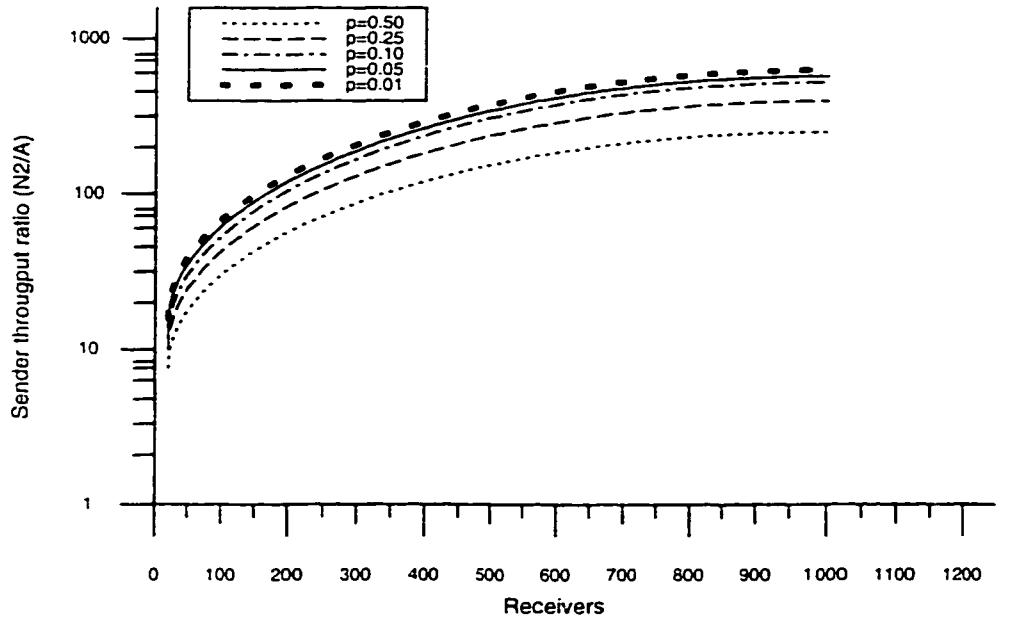

Figure 21: Sender Throughput Ratio (N2/A) vs. Number of Receivers

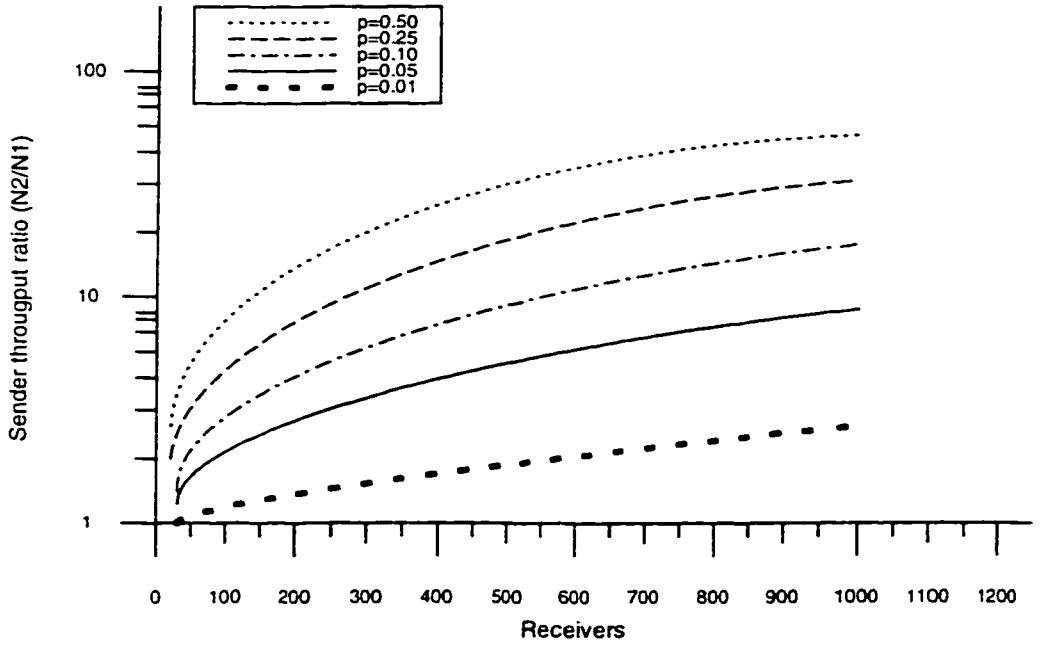

Figure 22: Sender Throughput Ratio (N2/N1) vs. Number of Receivers

As expected, broadcast NACK sender-oriented/receiver-initiated approach outperforms the sender-initiated approach. In addition, broadcast NACK approach performs better than the unicast NACK approach when the loss probability is high or when the number of receivers increases. That is due to light processing at the sender. With the broadcast approach, the sender needs only to process one NACK for each lost packet, instead of processing a NACK from each affected receiver.

Although the broadcast NACK approach may have the disadvantage of unnecessary bandwidth consumption in a low loss environment, it may generally be considered a better approach as it is more scalable than both sender-initiated and unicast NACK receiver-initiated approaches.

3.3.3.2 Flat Receiver-Oriented Error Recovery

The flat receiver-oriented approach allows receivers to communicate with each other for the purpose of error recovery. Whenever a receiver detects an error, it multicasts a NACK to the entire group. Each receiver caches data for some period of time or for the entire session. If any group member has correctly received the data, it issues a repair for the requested data. To avoid both NACK implosion and repair implosion, a receiver must delay itself for a random period of time before issuing a NACK or a repair. A NACK or a repair is only issued if the delay time expires before another receiver has issued a similar NACK or repair. In fact this random delay approach is not the only method to suppress NACK/repair requests. Deterministic suppression is also possible along a linear topology where downstream receivers detect the same errors as upstream receivers. By accurately estimating the delay between receivers, the timers of downstream receivers can be adjusted to produce longer delays. Thus, it is likely that the downstream receiver will observe the NACK of an upstream receiver before it issues its own NACK [Macker/Klinker, 1996].

Since most networks exhibit both linear and star (equidistant) characteristics, a combination of randomized and deterministic NACK/repair suppression should be used for a flat receiver-oriented reliability scheme [Macker/Klinker, 1996].

The bandwidth consumption of the flat receiver-oriented approach is comparable to the broadcast NACK sender-oriented approach. In both approaches, NACKs and repairs are broadcast globally to all members. This is considered as a drawback to the flat receiver-oriented approach since NACKs and repairs consume bandwidth for the whole group even for isolated packet losses [Macker/Klinker, 1996]. This problem, however, can be reduced by having the scope of the repairs be more localized.

In general, this approach is better than the Broadcast NACK sender-oriented. Both approaches provide the same scalability level, yet the flat receiver-oriented approach is fault tolerant since each receiver is capable of issuing repairs. The biggest disadvantage of the approach, however, is its data cache requirements on the receivers. In reality, this may not be considered as a real weakness for many applications, such as shared whiteboard, where data may have to be maintained in any event.

3.3.3.3 Hierarchical Receiver-Oriented Error Recovery

The hierarchical receiver-oriented approach is similar to the flat receiver-oriented approach in that receivers are allowed to communicate with each other for the purpose of error recovery. However, with the hierarchical approach, only designated receivers are allowed to assist in error recovery. With this approach the sender and the receivers are organized into a multicast delivery tree. The designated receivers form an error recovery hierarchy within the multicast tree.

In general, introducing a hierarchical structure into reliable multicasting increases scalability and allows for distributed state garbage collection and more organized repairing schemes [Macker/Klinker, 1996]. Various hierarchical error recovery algorithms have been proposed. One such algorithm is the *distributed logging* by Holbrook and Singhal, in which logging capabilities are added at client sites. Figure 23 illustrates the service architecture of distributed logging.

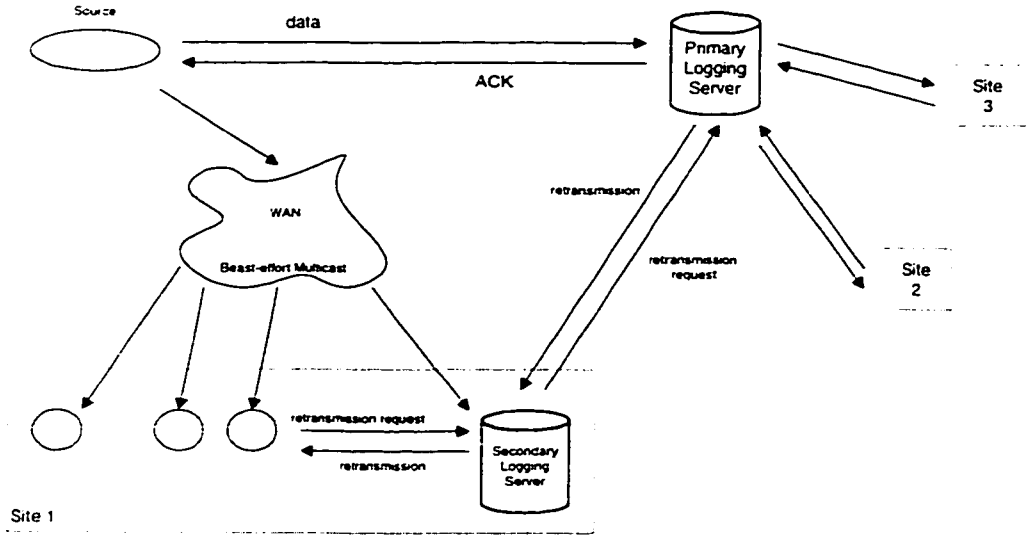

Figure 23: Distributed Logging Service Architecture

As shown in the figure, each site's logging server logs packets from the multicast source. Whenever a receiver detects a packet loss, it requests the packet from its secondary logging server, rather than from the source. If the secondary logging server does not have the requested packet, and can not retrieve it locally from other receivers in its site, it calls back the primary logging server for the lost packet. That way, only one retransmission request to the primary logging server is originated from each site. Moreover, the source only receives an acknowledgment from the primary server signaling that the packet is correctly received, and hence it can be discarded.

Macker and Corson introduced another hierarchical error recovery algorithm, in which a self-organizing shared repair tree is formed. Localized scoping in this approach limits repair dissemination and the hierarchy aids in buffer management [Macker/Klinker, 1996]. A similar approach is the Tree-based Multicast Transport Protocol (TMTP), which forms a source repair hierarchy. TMTP is discussed in more detail in the next chapter.

In general, the hierarchical error recovery approach is preferable over the flat receiver-oriented approach since requests for missing data and the corresponding repairs are limited to those network areas where the loss occurred. In addition, fewer receivers are burdened with assisting with error recovery than in the flat receiver-oriented approach. In fact, for some applications, such as time-sensitive applications where error recovery must be performed within a bounded amount of time, using a hierarchical error recovery approach might be most appropriate. Since the approach attempts to retransmit missing packets from a local retransmission agent instead of a possibly remote source, it reduces the latency of repairs compared to sender-initiated and receiver-initiated/sender-oriented approaches. The major disadvantage of hierarchical error recovery lies in the setting up and management complexity that it adds to the protocol. In addition, the approach has less flexibility than flat receiver-oriented since fewer receivers can assist with error recovery.

3.3.3.4 Absolutely Reliable Receiver-Initiated Error Recovery

To provide absolute reliability in a receiver-oriented approach, some specific constraints have to be imposed on the senders. Since senders are not tracking the receiver's receiving states, they must, strictly speaking, buffer the sent data indefinitely so that they can retransmit any requested packet to receiver at any future time. In a long-lived session, with a finite storage capacity, this constraint turns to be very problematic. However, for applications that require bounded latency reliability, this constraint may not be an issue, since expired data can simply be discarded.

To support absolute reliability, some form of ACKing mechanism from the receivers is required to allow the sender to periodically flush its buffers. This ACKing mechanism can be used in conjunction with the more general NACK error recovery approach and should be as infrequent as possible to reduce ACK implosion and general overhead. This mechanism also requires the sender to have the knowledge of the receiver set at any given time. Thus a scheme supporting absolute reliability represents a mixed requirement of both sender-reliable and receiver-reliable multicasting [Macker/Klinker, 1996].

3.3.4 Summary of Error Recovery Approaches

Which error recovery approach should be used, depends on both the application requirements and the availability of certain conditions. For example, retransmitted packets can either be multicast to the group or unicast to the requester. Unicasting a lost packet requires the sender or the designated receiver to have explicit knowledge of the receiver set. In fact, even if this knowledge can easily be available, multicast retransmissions might still be more appropriate for high loss networks where receivers share the same loss characteristics. Unicasting retransmissions could be appropriate for low loss networks and when failures are independent of each other. Tables 3 and 4 provide a general summary of both sender-initiated and receiver initiated error recovery approaches respectively.

Error Detection Method	Repair Method	Processing Load	Scalability
Receivers unicast ACK to the sender	Sender multicasts repairs	Sender: High for large receiver set Receiver: Low	Low: due to ACK implosion

Table 3: Characteristics of Sender-initiated Error Recovery

Receiver-initiated Error Recovery Method	Error Detection Method	Repair Method	Processing Load	Scalability
Unicast NACK Sender-oriented	Receivers unicast NACK to sender	Sender multicasts repairs	*Sender*: High, but less than sender-initiated *Receivers*: Low	Low, but better than sender-initiated
Broadcast NACK Sender-oriented	Receivers broadcast NACK to sender and all receivers	Sender multicast repairs	*Sender*: Less than unicast NACK sender-oriented *Receivers*: More than unicast NACK sender-oriented	Moderate
Flat Receiver-oriented	Receivers multicast NACK to sender and all receivers	Receivers cache data and are capable of issuing repairs	*Sender*: Low *Receivers*: More than sender-oriented	High, due to limiting NACK implosion
Hierarchical Receiver-oriented	Receivers NACK to a designated receiver	Designated receivers cache data and issue repairs	*Sender*: Low *Receivers*: Low *Designated receivers:* Higher than other receivers	Very high scalability and minimal network overhead

Table 4: Characteristics of Receiver-initiated Error Recoveries

3.3.5 Scalability

With the increasing popularity of multicasting, the issue of scalability has become one of the most important aspects related to reliable multicasting, and the subject of much cooperative research around the world. To what extent the number of receivers can be increased, while a protocol still provides reliable services with an acceptable performance, is difficult to evaluate. However, examining the major aspects that affect scalability can provide a possible estimation. These aspects include the bandwidth requirements and limitations, the processing amount needed at senders and receivers, the amount of receivers' reception state that the senders need to maintain, and the complexity of the algorithms needed to mange a large number of receivers. In fact, these aspects are mainly influenced by the error recovery scheme and the ordering guaranteed by a protocol.

3.3.6 Late-join and Early-leave Receivers

Since receivers may be allowed to join an in-progress session, or leave before the session has completed, a multicast protocol must be able to handle such situations. This may influence some other parts of the protocols that are related to other aspects such as error recovery, traffic load, protocol overhead and the amount of state maintained by the sender.

3.3.7 Successful Delivery

One aspect of protocol reliability is based on the ability to successfully deliver messages based on specific conditions. For example, for real-time applications, successful delivery of a message may be based not only on reliable delivery, but also on delivery within a specific bounded time from the moment the message was originated or possibly form the moment the message arrived at a specific member.

Another aspect of reliability is based on successful delivery to one or more members of the receiver set. Kasshoek [Kasshoek, 1992] defined the following alternatives: k-delivery, quorum delivery and atomic delivery. With k-delivery, reliability is assumed if at least k receivers successfully receive the message, for some constant k, that is less than the total number of receivers. With quorum delivery, the multicast operation is successful if the majority of receivers have successfully received the message, where the majority here is a specific percentage of the total number of receivers. Atomic delivery considers the multicast to be successful only if all receivers have successfully received the message.

Another semantic for a successful delivery may be based on specific k-delivery. That is, a multicast transmission is considered reliable if at least a specific set of the receivers has successfully received the message.

3.3.8 Flow Control

The way a protocol's flow control policies are implemented is very significant, since it can directly impact the protocol performance. As a matter of fact, controlling the packet rate in multicasting is more complicated than in unicasting because a multicast protocol must accommodate multiple receivers simultaneously. In addition, receivers within one session may be heterogeneous, with different access to hardware and network resources. The links to different receivers may also be of different capacity. Packets may be dropped because of poor quality links or network congestion. Although an increased number of retransmission requests may indicate either case, the protocol has to handle each situation in a different manner to appropriately adjust the transmission rate.

3.3.9 Ordering

Although ordering may not be applicable to many reliable multicast communications, certain applications do require some level of guaranteed ordering in addition to reliable delivery of messages. In general, ordering services can either be provided by the communication protocol – specifically, by the transport layer of the protocol, or alternatively by the application itself. Although providing the services at the transport layer places an extra burden on the protocol, it significantly simplifies the design and implementation of distributed software. In addition, it avoids performance degradation at the application level due to the execution of the additional ordering, synchronization and concurrency functions.

Again, the requirements for ordering differ broadly from one application to another. Some applications may not have any ordering requirements, while others may require some sort of absolute and strict ordering. Still, many other applications require only some sort of partial ordering. These different requirements by applications led to different classification of ordering such as: single source ordering, causal ordering, multiple source ordering, and total ordering.

3.3.9.1 Single Source Ordering

With single source ordering, all messages that are sent by a source to a multicast group must be delivered to all destination processes in the same order in which they have been sent. Guaranteeing the single source ordering is relatively simple and is sometime done by the underlying communication network [Garcia-Molina, 1991]. The basic idea is to have the sender to assign an ordered sequence number to each transmitted message. Receivers can then guarantee the correct order by passing messages to the application based on the sequence numbers. Having sequence numbers also allows a receiver to detect whether it has missed any messages.

3.3.9.2 Causal Ordering

Causal ordering concerns of ordering messages that are related to each other. Many applications, such as conferencing or workflow management applications, usually generate messages that are causally related [Petitt, 1996]. For example, users in a conferencing application may respond to an earlier message(s) by other user(s). With causal ordering the respond message by the user should not be delivered to any participant before all the related previous messages have been already delivered. Figure 24 illustrates an example of causal ordering. In the figure, A sends first a message m1 to B, C and D. C then responds to that message with message m2. Finally D responds to m2 by sending m3. With causal ordering, all of these related messages are delivered at all receivers in the order they have been transmitted, that is (m1, m2 , m3).

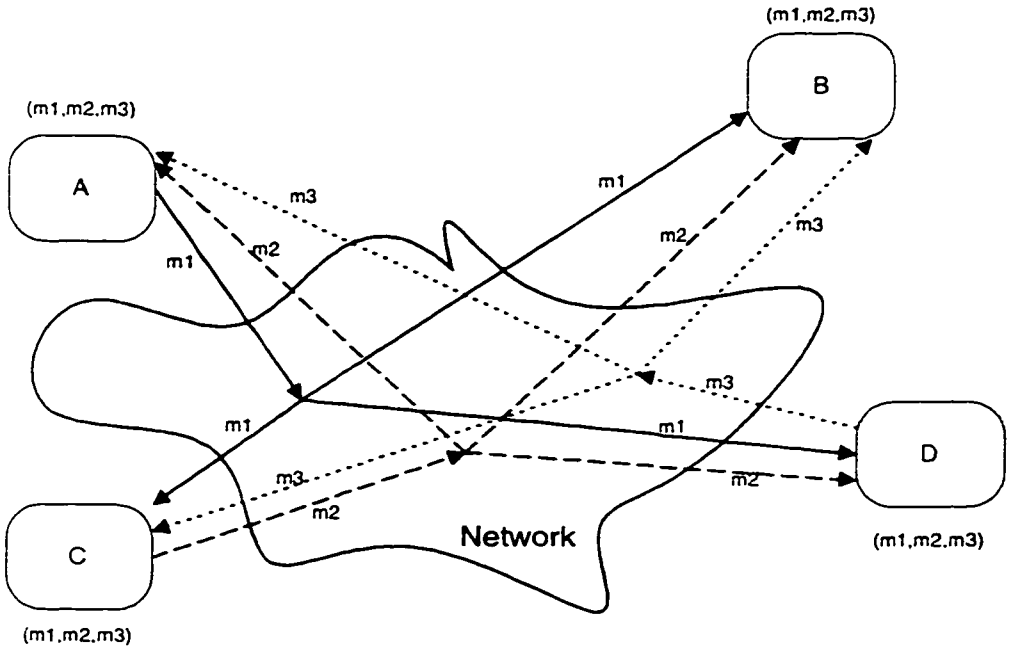

Figure 24: Example of Causal Ordering

3.3.9.3 Multiple Source Ordering

Multiple source ordering is similar to, but stricter than, causal ordering. With this ordering method, if messages m1 and m2 are transmitted, either by same or different sources, to a multicast group then all destination processes must get them in the same relative order regardless whether these messages are related or not.

3.3.9.4 Total Ordering

With total ordering, messages have to be delivered in the same relative order in which they have been transmitted, even if they are sent by different sources and are addressed to different but overlapping multicast groups. This level of ordering may be required for specific applications such as distributed databases. For example, if two sources update then send a copy of a database, then the update that is sent first should be received first at all the receivers, followed by the update that was sent later. If this is not the case, it is very possible that different receivers will end with inconsistent copies of the database. Total

ordering can be used in such circumstances to eliminate such problems. Total ordering, however, has a major disadvantage, which is its performance implication on the protocol due to the delay time needed to achieve ordering. Figure 25 illustrates an example of total ordering. In the figure, sender S1 sends an update U1 at time T1, then sender S2 sends an update U2, of the same information, at time T2. With total ordering all the receivers must receive the updates in the same order that they were sent; that is (U1, U2).

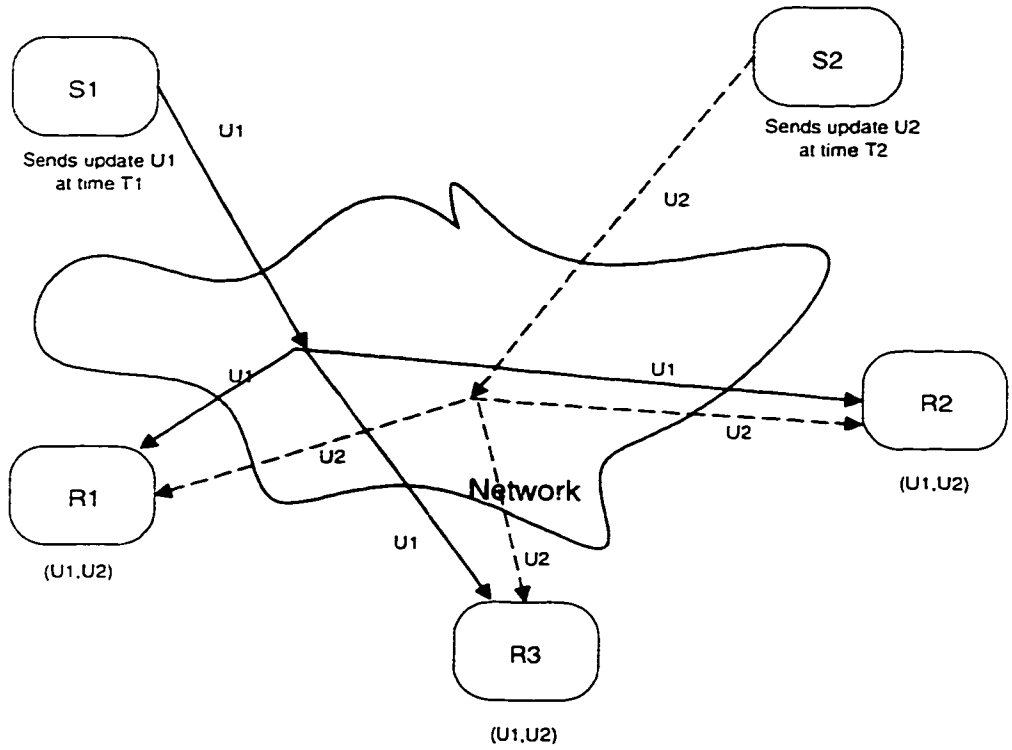

Figure 25: Example of Total Ordering

Many different solutions have been proposed to achieve total ordering. One solution recommended the use of real timestamps to enable the receivers to order packets properly. In this solution, senders attach a synchronization message to each message they multicast. This synchronization message includes mainly a timestamp in addition to other information. With this synchronization information, each receiver is capable of ordering the received messages properly, based on a local clock, before delivering the messages to the application.

Another solution proposed by Garcia-Molina is based on the construction of a propagation graph. In this solution, senders do not multicast messages to all the receivers. Instead, they transmit their messages to one central receiver. The central receiver then orders all the messages properly before transmitting them to the rest of the receivers. Figures 26a and 26b show an example of this solution. Figure 26a shows a transmission scenario where ordering is needed, while figure 26b shows how ordering can be achieved via constructing a propagation graph. Garcia-Molina also provided a variant of this solution that is based on the use of a collection of a few nodes to provide the needed ordering instead of depending on a single central node for that.

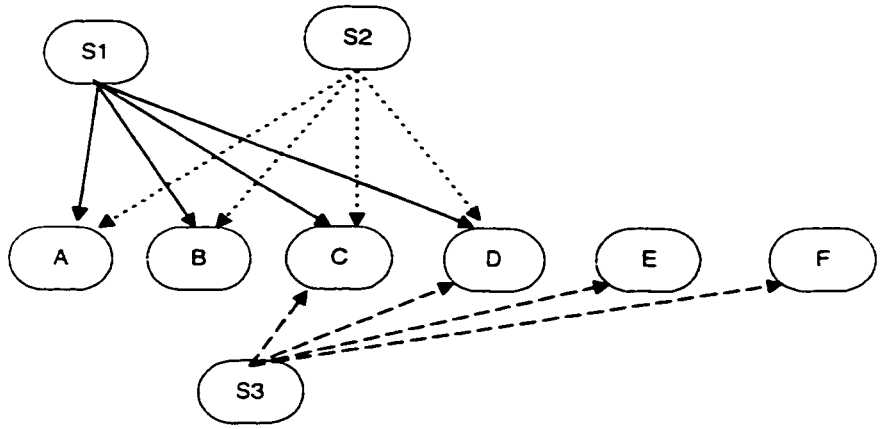

Figure 26a: Example of a Multicast Transmission where Ordering is Required

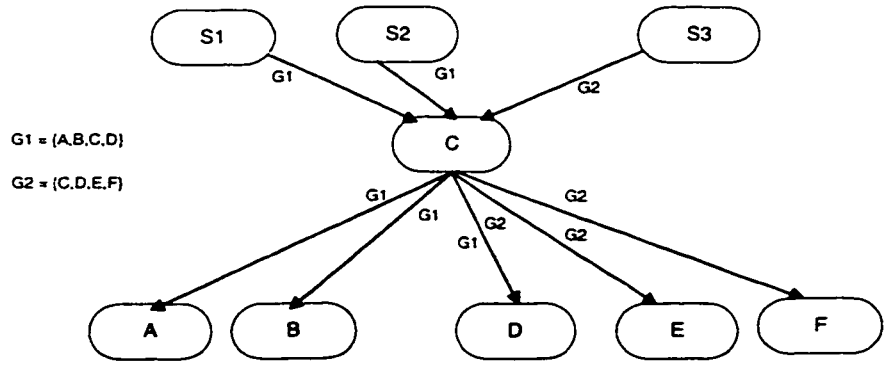

Figure 26b: Constructing a Propagation Graph to Achieve Ordering

The solution provided by Garcia-Molina might be considered a better approach than the simple timestamp solution described above, since the delay at a receiver is dependent only on the connection between it and the central receiver. With the timestamp approach, one receiver with a large network delay between itself and the sender may cause the synchronization delay of all other receivers to be increased.

3.3.10 Failure Recovery

Failure occurrences are very possible due to many reasons such as link failure, congestion, data overflow, or some other factors. The way a reliable multicast protocol responds to failures is very significant. Different failures may need to be handled in completely different fashions. Moreover, the protocol may need to handle a single failure in different ways depending on the existing circumstances at the failure time as well as after the failure occurrence. That is, after a failure, the protocol may be able to fully recover all lost data, and resume its services as expected, or it may need to make many adjustments. For example, if the failure occurred due to a permanent failure of one or more links within the network, then the protocol may need to reform the different multicast groups to avoid the failed link. It is still up to the application then whether to accept or reject to continue with the applied adjustments.

It is also very possible that a protocol may not be able to provide all its expected services after a failure. Garcia-Molina gave an example of such situation for a protocol that provides reliable and ordered delivery multicast services. After a failure, the level of ordering guaranteed by the protocol can widely differ as follows:

- No order guarantees: ordered delivery is no longer guaranteed at any site.
- Consistent delivery: messages are delivered in consistent order only at operational sites. Should any failed site come back, no recovery against missed messages is to be provided to that site. Additionally, the failed site message history before the failure becomes irrelevant.
- Rollback for recovery: sites that recover from the failure are forced to rollback and redeliver messages that they have inconsistently delivered. In such a case, the failed sites may temporarily deliver

messages in an incorrect order. However, they eventually should compensate for this and also recover any missed messages.

- Strict Consistency: ordering is strictly guaranteed. Inconsistent delivery is never allowed.

Surely, some other classifications may still hold. The term "eventually", for example, may be the subject of an entirely different spectrum of reliability.

3.4 Conclusion

For many multicast applications reliability is essential. However, the definition of reliability may differ broadly from one application to another. For example, for some applications reliability may be interpreted as guaranteed delivery of multicast messages within a maximum amount of time. For other applications reliability may be interpreted as guaranteed delivery of messages to a specific set of members, regardless of the amount of time it takes to deliver the messages. Yet, for other applications the order in which messages are delivered may mainly drive the definition of reliability of those applications.

Once reliability is precisely defined, a protocol can start implementing the needed reliability functionality. Ideally for the applications, all the needed reliability functionality should be implemented at the lower layers of the protocol stack. This would allow the application to worry only about its own implementation rather than how reliability is guaranteed. Unfortunately, in reality this may not be feasible since parts of the implementation of the reliability functions may need to be done at the higher levels. For example, for applications that demand end-to-end reliability, such as file transfer applications, parts of the implementation must be done at the higher layers. Additionally, some other applications, such as finance and banking applications, prefer to perform their own error and integrity checking, regardless of the level of reliability guaranteed by the protocol. In such cases, having all the reliability functions implemented at the higher layers would only produce unnecessary delays. To conclude, it is better to move some of the reliability functions to the higher layers of the protocol stack, instead of having all of them implemented at the lower layers.

Another major issue of reliable multicasting concerns error recovery. Two major error recovery approaches exist – the sender-initiated and the receiver-initiated. In addition, different receiver-initiated error recovery methods exist – unicast NACK sender-oriented, broadcast NACK sender-oriented, flat receiver-oriented and hierarchical receiver-oriented.

The sender-initiated error recovery approach may be used when the sender is capable of having a full knowledge of the receiver set. However, using this approach limits the scalability level of the protocol since a high number of receivers may cause the sender to be imploded.

To allow the protocol to become scalable, a receiver-initiated error recovery approach is to be utilized. The unicast NACK sender-oriented approach allows the least level of scalability among all receiver-initiated error recovery methods, while the hierarchical receiver-oriented provides the highest level of scalability. In all cases, all receiver-initiated error recovery methods provides a higher scalability level than the sender-initiated approach.

To conclude, since reliability is a major concern of today's applications, the hierarchical receiver-oriented approach is considered to be the best approach for error recovery. The sender-initiated approach should only be used when the application requires a very high level of reliability and for sure when the sender is capable of having a full knowledge of the receivers.

Other important reliable multicast issues are ordering, late-join and early-leave receivers, flow control and failure recovery. Although not all of these features may be required by all applications, any protocol that is designed with the goal of becoming a standard reliable multicast protocol must provide all of these features. It is then up to the application to turn any of these features on or off.

Chapter 4

Reliable Multicast Protocols

During the last few years, many reliable and semi-reliable (best effort) multicast protocols have been designed and implemented. Unfortunately, no standard has been realized yet for reliable multicast protocols. In fact, many of the existing protocol were merely designed to serve specific applications, or to achieve specific functionality, such as ordering or scalability.

In this chapter, we examine several major existing reliable multicast protocols. These protocols are: the Tree-based Multicast Transport Protocol (TMTP), the Scalable Reliable Multicast (SRM), the Reliable Multicast Transport Protocol (RMTP), the Reliable Adaptive Multicast Protocol (RAMP), the Reliable Multicast Protocol (RMP), the Multicast Transport Protocol (MTP-2), the Local Group based Multicast Protocol (LGMP), and the Xpress Transport Protocol (XTP). The chapter examines the different aspects of each protocol according to the taxonomy described in chapter 3, and provides a theoretical evaluation of the protocol. If real analysis has been conducted for a protocol, the results are also included. Protocols' description, evaluation and real analysis are given in detail in [Armstrong/Freier, 1992], [Atwood, 1996], [Bormann/Ott, 1994], [Callhan/Montgomery, 1996], [Floyd/Jacobson, 1996], [Hofmann, 1996], [Koifman/Zabele, 1996], [Lin/Paul, 1996], [Montgomery, 1994], [Ott/Bormann, 1994], [Paul/Sabnani, 1996], [Petitt, 1996], [XTP Forum, 1995], and [Yavatkar/Griffioen, 1995].

4.1 The Tree-based Multicast Transport Protocol (TMTP)

TMTP is a reliable multicast protocol that is designed for multimedia applications that involve a large number of participants, where participants are additionally allowed to dynamically join or leave the application. TMTP utilizes the efficient best-effort delivery mechanism of IP multicast for packet routing and delivery. However, for the purpose of scalable flow and error control, the protocol dynamically organizes the participants into a hierarchical control tree. The control tree employs restricted NACKs with

suppression, and utilizes an expanding ring search to distribute the functions of state management and error recovery among many members, thereby allowing the number of receivers to scale well [Yavatkar/Griffioen, 1995].

In general, receiver groups in the hierarchy are constructed in such a way that members in the same subnet are assigned to a domain. A single domain manger then acts as a representative on behalf of the members of that domain.

4.1.1 Packet Transmission

TMTP provides one-to-N multicast: that is, one sender is only allowed within a communication session. The sender multicasts a stream of information to a *dissemination* group. A dissemination group consists of processes scattered throughout the Internet, all interested in receiving the same data feed. Once the dissemination group has been constructed, the sender can start multicasting data to the group. The multicast packets travel directly to all group members using standard IP multicast. In addition, all the domain managers in the control tree listen and receive the packets directly as well.

In order for the sender to reclaim buffer space and implement flow control, receivers have to send back positive acknowledgments once they receive the data successfully. However, to avoid implosion, the sender receives acknowledgments only from its immediate domain managers, rather than directly from all the receivers. Since TMTP uses a local recovery approach, a domain manager can immediately send an acknowledgment to the sender for each packet that it receives. Should any member within that domain miss the acknowledged packet, the domain manager retransmits the packet to it. To reduce implosion even more, domain managers send the ACKs to their parents only periodically. This feature substantially reduces ACK processing at the sender and at each domain manager.

4.1.2 Error Control

TMTP uses a combination of sender-initiated and receiver-initiated error recovery approaches. The sender requires periodic positive acknowledgements from its immediate domain managers. Similarly, domain mangers require positive acknowledgements from their group members or their children domain managers. If the sender (or a domain manger) does not receive an acknowledgment within a specific amount of time, it assumes that an error has occurred and hence issues a retransmission. In addition, TMTP uses restricted NACKs with NACK suppression to respond quickly to packet losses. Once a receiver detects a packet loss, it generates a negative acknowledgment to be multicast to its domain manger and its entire group. However, to suppress NACK implosion, the receiver delays itself randomly before sending the NACK. If within that delay time, another NACK is heard for the same packet, the receiver cancels its request; otherwise the NACK is sent.

Since domain mangers are allowed to retransmit packets, and receivers are also allowed to multicast NACK requests, TMTP uses a technique called limited scope multicast messages to prevent messages targeted to a particular region of the tree from propagating throughout the entire Internet. This is implemented by setting the TTL value in the IP header to some small value, called the multicast radius. The appropriate multicast radius to use is obtained from the expanding ring search that domain managers use to join the tree [Yavatkar/Griffioen, 1995].

4.1.3 Flow Control

TMTP achieves flow control by using a combination of rate-based and window-based techniques. The rate-based component of the protocol prohibits senders and domain mangers from transmitting data faster than some predefined fixed maximum transmission rate. This maximum rate is set once the group is created. The rate is fixed to help avoid congestion arising from bursty traffic and packet loss at rate-dependent receivers [Yavatkar/Griffioen, 1995].

TMTP window-based flow control slightly differs from the conventional point-to-point window-based flow control. With the conventional method, the lower edge of the sliding window is advanced once an acknowledgment is received. If no acknowledgment is received within a specified amount of time, the packet corresponding to this acknowledgment is retransmitted. In TMTP, retransmissions are relatively expensive, since they are multicast. Additionally, transient traffic conditions or congestion in one part of the network can put a backpressure on the sender, causing it to slow the data flow. The protocol avoids these problems by partitioning the window, and by delaying retransmissions as much as possible. Two different timers are used to control the window size and the rate at which the window advances. The first timer defines a timeout period that begins when the first packet in a window is sent. The second timer defines a periodic interval at which each receiver is expected to unicast a positive ACK to its parent [Yavatkar/Griffioen, 1995]. This window partition along with delaying retransmissions increase the chance for a positive acknowledgment to be received and also allows domain managers to rectify transient behavior before it begins to cause backpressure and affect other parts of the control tree [Yavatkar/Griffioen, 1995].

4.1.4 Experimental Analysis

Different experiments were conducted across a WAN of seven sites distributed across the United States and Europe. The experiments measured the performance at each of these sites; each site performs a separate domain. All the experiments were conducted using standard IP multicast across the MBONE and thus experienced real Internet delays, congestion, and packet loss [Yavatkar/Griffioen, 1995].

As a point of comparison, the implementers of TMTP's implemented a standard sender-initiated transport protocol both with and without window-base flow control; these were referred to as the WIN_BASEP and the BURST_BASEP protocols [Yavatkar/Griffioen, 1995].

In general, two kinds of tests were conducted to find out both impacts of data size and group size on the protocol. TMTP was compared to two BASEP protocols. The processing load was evaluated at the sender and domain managers. The end-to-end delay was obtained when the number of receivers was 30, while the

size of the file being transferred was 3KB first and 30KB later. Similar tests were performed with the file size fixed to 30KB, while the number of receivers in a group varied from 5 to 30.

When the file size was increased, the processing load in both BASEP protocols did significantly increase, while in TMTP it increased only slightly. That is because the work was distributed among many nodes in the control tree, so that the sender did not have to process acknowledgments or retransmission requests from all the receivers. End-to-end delay was increased in all protocols when the file size was increased. However, the increase in TMTP occurred at a significantly lower rate than in both the BASEP protocols.

When the group size was increased, both BASEP protocols experienced a sharp increase in the processing load, while TMTP experienced almost no increase. That is because the processing load at the sender is limited by the maximum number of immediate children in the control tree. However, the processing load is expected to increase with a higher number of domains, since the domain mangers have to adopt more children [Yavatkar/Griffioen, 1995].

End-to-end delay also increased significantly with both BASEP protocols compared to TMTP. The primary reason for this difference stems from TMTP receiver-initiated capabilities, which respond to and correct errors quickly. In contrast, the BASEP protocols do not correct an error until a retransmission timeout occurs [Yavatkar/Griffioen, 1995].

A final important analysis of the protocol performance would evaluate its bandwidth consumption. Unfortunately, it was hard to quantify the amount of bandwidth consumed by each protocol. However, the results indicated that TMTP generated far fewer retransmissions than the BASEP protocols, and most TMTP retransmissions were local to a particular domain [Yavatkar/Griffioen, 1995]. This could be considered as a rough comparison of bandwidth consumption.

4.1.5 Protocol Evaluation

4.1.5.1 Strength

The hierarchical structure of TMTP along with its error recovery technique allows the protocol to provide both complete reliability and scalability. The use of receiver-initiated error recovery within a domain reduces the processing load on domain mangers. The use of sender-initiated error recovery allows the sender and domain mangers to reclaim their buffer space upon receiving the ACKs. The hierarchical structure significantly reduces the processing burden on the sender as well as other members of the tree, since the load is distributed.

4.1.5.2 Weakness

The approach used by TMTP to select domain managers and organize group members is solely based on a TTL value. A new domain manager attaches to the control tree after discovering the closest node in the tree, using an expanded ring search [Yavatkar/Griffioen, 1995]. This method may easily result in assigning members with very different error characteristics to the same group [Petitt, 1996]. This may cause the retransmission operations to become inefficient, since receivers with bad error characteristics would affect the rest of the receivers in the group.

Under specific special circumstances, the protocol may not be able to provide the required reliability. For example, if a domain manager wishes to leave the communication after its last local member has left, then all its child domain mangers have to be reintegrated into the tree. With the current implementation of TMTP, any errors that arise during the brief reintegration time might not be recoverable [Yavatkar/Griffioen, 1995].

Since the control tree is built solely at the transport layer, some additional end-to-end delay may be introduced due to any inefficient packet routing.

4.2 The Scalable Reliable Multicast Protocol (SRM)

SRM is a reliable, M-to-N multicast framework, which has been designed for light-weight sessions and application level framing. The algorithms of this framework are efficient, robust, and scale well to both very large networks and very large sessions [Floyd/Jacobson, 1996]. SRM has been prototyped in the distributed whiteboard application (*wb*). The design of SRM is based on the Application Level Framing (ALF) concept. ALF says that that the best way to meet diverse application requirements is to leave as much functionality and flexibility as possible to the application. Therefore SRM is designed to meet only the minimal definition of reliable multicast; that is, eventual delivery of all the data to all the group members, without enforcing any particular delivery order [Floyd/Jacobson, 1996].

4.2.1 The Distributed Whiteboard (wb)

wb is a network conferencing tool that is designed based on the ALF principle. *wb* provides a *drawing surface*, where users write/draw their information. *wb* separates the drawing into pages, where a new page corresponds to a new viewgraph in a talk, or the clearing of the screen by a member of a meeting. Any member can create a page and any member can draw on any page [Floyd/Jacobson, 1996]. A globally unique identifier, the Source-ID, identifies each member. Each page is identified by a Page-ID, which consists of the Source-ID of the initiator of the page and a unique page number local to that initiator. Each drawing on the whiteboard produces a stream of drawing operations, or "drawops", that are timestamped. In addition, each drawop is assigned a sequence number that is relative to the sender.

wb has no requirement for ordered delivery of drawops. That is, drawops are ordered by the application itself and not by the multicast protocol. Receivers use the timestamps assigned to drawops to determine the order of operation's appearance on the whiteboard. This allows wb to provide the needed ordering without adding more complexity or extra delay to the protocol.

wb gives more priority to repair requests for the current page over new data. That is, whenever a member is allocated bandwidth to transmit data, the highest priority goes to repairs for the current page, then to new data, and finally to repairs for other pages.

4.2.2 Session Messages

Originally, session messages were proposed for reliable multicasting to enable receivers to detect the loss of the last packet, and to enable the sender to monitor the receivers' status. In SRM, each member multicasts periodic session messages that report the sequence number state for active sources. Additionally, members use session messages to determine the current participants of the session, and to estimate the one-way distance between nodes [Floyd/Jacobson, 1996]. In average, the bandwidth consumed by session messages is limited to a small fraction (e.g., 5%) of the aggregate data bandwidth.

4.2.3 Error Recovery

SRM uses flat receiver-oriented receiver-initiated error recovery. When a receiver detects a hole in the data stream, it multicasts a repair request to the entire group. However, to avoid implosion as well as sending identical requests for the same data, SRM uses the slotting and damping mechanisms, which have been introduced by the Xpress Transport Protocol (XTP). Receivers delay themselves randomly before sending a repair request, and refrain from sending the request if a similar request is sent within the delay time. As with the original data, repair requests and retransmissions are always multicast to the entire group. To avoid transmitting duplicate repairs, hosts that wish to wish to transmit a repair delay themselves randomly before transmitting the repair. The time delay is a function of the distance in seconds between the member that wishes to send the repair and the one which triggered the request or repair [Floyd/Jacobson, 1996]. Thus, it is more likely that a closer host to the point of failure is to timeout first and multicast the request.

Reliable data delivery in SRM is ensured as long as each data item is available from at least one member [Floyd/Jacobson, 1996]. This has the advantage of reducing the buffering requirements on all members within the communication session.

4.2.4 Congestion Control

The SRM basic framework for congestion control assumes that the members of the multicast session have an estimate of the available bandwidth for the session, and constrain the data transmitted to be within this estimated bandwidth [Floyd/Jacobson, 1996]. However, this framework does not seem to be very precise since it does not indicate how members can determine this available bandwidth. Additionally, there is no indication of how a congestion situation can be detected or how to avoid potential congestion. Thus, Floyd and Jacobson have proposed some other possible congestion control approaches for SRM.

4.2.5 Experimental Analysis

Simulations have been conducted to examine the performance of the loss recovery algorithms [Floyd/Jacobson, 1996]. The simulations showed that the loss recovery algorithms perform well when many of the nodes in the underlying tree are members of the multicast session. However, the loss recovery algorithms performed less well for a sparse session with a large tree [Floyd/Jacobson, 1996]. That is, the algorithms performed less efficiently when the session size is relatively small compared to the size of the underlying network, and when members are scattered throughout the net [Floyd/Jacobson, 1996]. This analysis motivates the development of the adaptive loss recovery algorithm, in which repairs and request timers are adjusted based on past error recovery performance. Simulations showed that using the adoptive loss recovery algorithm in various situations resulted in a quick reduction in the number of duplicates.

4.2.6 Protocol Evaluation

4.2.6.1 Strength

The obvious strength of SRM is its ability to provide M-to-N multicasting. The protocol allows any member within the communication session to multicast data to the rest of the members. This feature could be required by many applications.

The adaptive loss recovery algorithm allows applications using the SRM framework to adapt to a wide range of group sizes, topologies and link bandwidths while maintaining robust and high performance [Floyd/Jacobson, 1996]. In other words, the adaptive loss recovery algorithm improves the scalability of the flat receiver-oriented receiver initiated error recovery approach.

4.2.6.2 Weakness

SRM does not provide any concrete method for congestion control. The basic framework for congestion control is very imprecise and unreliable.

SRM provides just a certain level of reliability. The protocol meets a minimal reliability definition of delivering all data to all group members, deferring more advanced functionality, when needed, to individual applications [Floyd/Jacobson, 1996].

SRM does not provide guaranteed ordered delivery of data. Ordering is to be totally provided by the application if needed. This may not be considered as a weakness for many applications where ordering is not required.

Although the adaptive loss recovery algorithm improves the scalability of the flat receiver-oriented receiver initiated error recovery approach, the algorithm has a problem, normally referred to as the "crying baby"

problem. If a single link to one member of the group has a high error rate, then all members of the multicast group will contend wit a multicast request and one or more multicast responses. A member of the multicast group connected by a wireless link or a congested link will result in "crying baby" problem [Paul/Sabnani, 1996].

4.3 The Reliable Multicast Transport Protocol (RMTP)

RMTP provides sequenced, lossless delivery of bulk data from one sender to a group of receivers. RMTP is based on a hierarchical structure in which receivers are grouped into local regions or domains. The hierarchy is rooted at the sender, while each group is rooted at a special receiver called the Designated Receiver (DR). The DR (in particular a specific component of the DR called the Acknowledgment Processor AP) is responsible for processing ACKs from its group members and sending them periodically to the sender, as well as for retransmitting lost packets to receivers in its domain. RMTP can be built on top of either virtual-circuit networks or datagram networks. The only service expected by the protocol from the underlying network is the establishment of a multicast tree from the sender to the receivers [Paul/Sabnani, 1996].

4.3.1 Session Manager

RMTP relies heavily on a session manager (SM) for performing specific functions. The session manger is not part of RMTP, but it is used at the session layer to mange an RMTP session. The session manger is responsible for providing many functions such as supplying connection parameters between sender and receivers, detecting and handling errors in case of network partitions or crash, and choosing and setting the maximum transmission rate.

4.3.2 Packet Transmission

RMTP design is based on IP-Multicast strategy in which the sender does not explicitly know who the receivers are. Receivers may also join or leave at any time within the session without informing the sender. Once the session manger has provided the sender and the receivers with the session information, the sender can start multicasting packets to the receivers. The transmission occurs at regular intervals that are determined by the session manager. However, the number of packets transmitted within each interval may vary depending on the space available on the send window at the sender. At most, the sender can transmit one full window of packets during a time interval. This controls the sender's maximum transmission rate. Additionally, during network congestion, the sender is also limited by a congestion window within the same time interval [Paul/Sabnani, 1996].

4.3.3 Error Recovery

RMTP uses receiver-initiated error recovery. The sender assigns a sequence number to each transmitted packet. Each receiver periodically informs the sender of its status by sending ACKs. An ACK consists of two parts, a sequence number and a bitmap. The sequence number indicates the sequence number of the last received packet. The bitmap identifies whether a packet was successfully received (indicated by bit 1) or not (indicated by bit 0). Figure 27 shows an example of a receiver's receive window. An ACK from this receiver would include a sequence number of 22 and a bitmap of 01101000 indicating that the last received packet has the sequence number 22, while only packets with sequence numbers 16, 17 and 19 were correctly received.

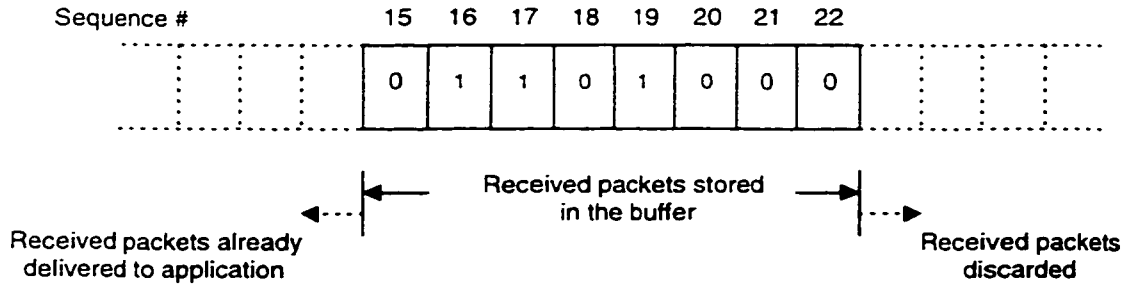

Figure 27: Example of an RMTP Receiver's Receiving Window

To avoid ACK implosion at the sender, receivers do not sends their ACKs directly to the sender. Rather, each receiver sends its ACK only to its direct DR. DR's then inform their upstream DR's of missing packets through the same acknowledgment mechanism.

How often a receiver should send ACKs is dynamically configured based on the round trip time between the receiver and its DR. If the time interval between ACKs is very lengthy, then it will take a long time for a receiver to get its needed retransmissions, which in turn will delay the delivery to the application. On the other hand, if the interval is very small, the DR may end up retransmitting the same packet multiple times without knowing if the first retransmitted packet was received correctly by the receivers [Lin/Paul, 1996].

The sender uses selective repeat retransmissions. Repairs may either be multicast to the DR's or be unicast to specific ones, depending on the number of requests for a packet. That is, if the number of requests exceeds a specific threshold, the sender multicasts the packet to all DR's including those ones that did not request the packet. The DR's uses the same approach when they retransmit the packets to receivers in their groups.

4.3.4 Late Joining Receivers

In RMTP receivers are allowed to join the application at any time. RMTP includes two features that allow late joining receivers to catch up with the rest – the immediate transmission request and the data cache.

4.3.4.1 Immediate Transmission Request

When a receiver joins an ongoing session, it checks the sequence number of the packet that is transmitted at that time. It then sends a special ACK to its DR. Once the DR receives this ACK, it checks the bitmap and unicasts all missed packets to the receiver.

4.3.4.2 Data Cache / Two-level Caching

For a DR to be able to provide all missed packet to a late joining receiver, it must buffer the entire file during the session. In a long-lived session, this may require each DR to have an infinite buffer, which is infeasible. To solve this problem, RMTP uses a two-level caching mechanism [Lin/Paul, 1996]. That is, the most recent packets up to the cache size are kept in memory, while the rest of the packets are stored in disk.

4.3.5 Flow Control

Since the sender in RMTP does not know either the identity of the DR's or how many of them exist at any point of time, it is not possible to use a simple window-based flow control mechanism. Such a mechanism would require the sender to get ACKs from all first level DR's before it can advance its window. To overcome the problem, the sender operates in cycles to advance its window. In the first cycle, the sender transmits a window full of new packets. In the beginning of the next cycle, the sender updates the send window and transmits as many new packets depending on the available space in its send window [Paul/Sabnani, 1996]. The window update is done as follows. The sender assumes only the existence of those DR's which have sent status messages to it within a given time interval. Once the sender receives

ACKs from those specific DR's indicating that the relevant packets were successfully received, it advances its lower window.

4.3.6 Congestion Control

RMTP uses retransmission requests from receivers as an indication of possible network congestion. The sender simply computes the number of ACKs with retransmission requests within a specific time interval. If this number exceeds a threshold, then the sender assumes a congestion situation and starts using a congestion window to reduce the transmission rate. The sender starts by setting the size of the congestion window to 1, which reduces the packet transmission to one packet per time interval [Paul/Sabnani, 1996]. The sender then recalculates the number of retransmissions, and adjusts the size of the congestion window accordingly. This method of congestion control is referred to as *slow-start* congestion control.

4.3.7 Experimental Analysis

RMTP's performance was measured by running tests over a WAN consisting of 18 receivers located at 5 geographic areas [Paul/Sabnani, 1996]. Four sites were located at the United States, and one site was located in Taiwan. A simplified version of the session manager was implemented for the experiment. Each site was served by one DR. The experiments performed multicast file transfer in three different network environments: a LAN, a domestic WAN, and a mixture of these environments. For the LAN test, only throughput, retransmissions and the number of slow starts were measured. For the other two tests, the number of duplicates was also measured. The most significant results can be summarized as follows.

The number of retransmissions for both WAN tests confirms the major rule that the DR's play in caching received data, processing ACKs, and handling retransmissions. In fact the DR for the Taiwan site transmited more packets to its area than the sender did to all areas. That is, without the DR the sender would have to handle these retransmissions.

The difference between the minimum and the maximum throughput in all measurements was very small. This indicated that receivers took about the same time to correctly receive the file, regardless of their geographic location, which means that RMTP is able to adapt to receivers in various network environments.

The number of duplicates was low at most of the sites. This implies the effectiveness of the algorithm used to calculate the ACK interval [Lin/Paul, 1996].

The number of slow-starts increased slightly when the tests were shifted from LAN to domestic WAN. Another slight increase was observed when the tests shifted to international WAN. This indicated that RMTP design has achieved its goal of not over-running slow receivers and not wasting network bandwidth. The result additionally showed suboptimal throughput for fast receivers.

4.3.8 Protocol Evaluation

4.3.8.1 Strength

The hierarchical structure of RMTP allows the protocol to provide a high degree of scalability. Having receivers geographically far from the sender does not represent a problem, since they are served by a DR, rather than by the sender. The state information kept at the sender is also independent of the number of receivers, which is a key to scalability [Paul/Sabnani, 1996].

RMTP is able to efficiently handle receivers in heterogeneous environments [Paul/Sabnani, 1996]. In particular, receivers in a relatively lossy network, such as wireless/congested network, can be made into a local region. The DR of that region will then be responsible for handling ACKs and retransmissions in that region [Paul/Sabnani, 1996]. This avoids other receivers within the communication session from being affected by the lossy region.

The ability of the sender and DR's to choose between multicasting repairs to a group or unicasting them to individual receivers is advantageous. With this flexibility, it is relatively safe to assign members with different error characteristics to the same group since a receiver with poor error characteristics may get all its retransmission requests using unicast transmissions from the DR without affecting the rest of the receivers in the group. This additionally reduces the number of retransmissions sent to receivers that do not need them.

4.3.8.2 Weakness

Since receivers determine their DR based on the TTL value, it is very possible that a large number of receivers choose the same DR [Paul/Sabnani, 1996]. This situation may result in load unbalancing among the different DR's.

The method used to detect and handle congestion is imperfect. The number of retransmission may be increased due to network congestion, or possibly due to poor quality links that cause more packets to be dropped. Although the number of retransmission requests can be used as an indication to either situation, a different handling method is needed in each case. If the network is congested, then reducing the transmission rate is appropriate. However, if the increased number of retransmission requests was caused by poor links, then the transmission rate should be increased to compensate for dropped packets [Petitt, 1996]. Thus, RMTP behaves correctly only if the network is really congested.

Some additional overhead is placed on the receivers and designated receivers since they have to periodically compute the round-trip time delay to determine how often they should be sending status messages. If this operation were not done dynamically, then the protocol performance would be affected. This is a trade-off between the performance of the protocol and the overhead in computing the round-trip time dynamically [Paul/Sabnani, 1996].

4.4 The Reliable Adaptive Multicast Protocol (RAMP)

RAMP is a transport layer multicast protocol that provides fully reliable point-to-multipoint transmission. The protocol is fully reliable in the sense that it is both sender and receiver reliable. In addition, the sender has a full knowledge of the receiver set. The application is notified if either a sender or a receiver fails. This feature is needed by applications where each user's application acts as a sender and a receiver to a single multicast group [Koifman/Zabele, 1996]. RAMP guarantees reliable and orderly delivery to all multicast recipients using immediate, receiver-initiated NACK-based unicast error notification combined with originator based unicast transmission.

RAMP features include timely notification of receiver failure to the sender, as well as timely notification of sender failure to all receivers. In RAMP, receivers are allowed to leave or join the application at any point within the session.

Since RAMP maintains explicit group membership at the sender, transmission can be terminated immediately when the last receiver leaves the group, so it does not consume valuable network resources unnecessarily.

4.4.1 Design Influences

RAMP has been designed for applications running over an all-optical circuit-switched, gigabit network under the ARPA-sponsored TestBed for Optical NEtworking (TBONE) project. The approach was motivated by the three loss characteristics of high performance networks, such as the TBONE. Firstly, the extremely low bit-error rates (10^{-12} or better), which leads to fewer retransmissions, and hence reduces network congestion. Secondly, the use of optical crossbar switches, which further reduce the chances of congestion, since they do not perform the usual store and forward packet operation. Finally, receivers in these networks are generally much faster than in conventional networks, so data processing is done quite fast. These characteristics make packet losses almost entirely a result of receiver buffer overflows rather

than network congestion. In addition, since receiver losses are highly independent, RAMP uses unicasting rather than multicasting for NACKs, and possibly for retransmissions. This has the advantage of eliminating unnecessary receiver processing overhead associated with reading and discarding redundant packets [Koifman/Zabele, 1996].

4.4.2 Passive and Active Opens

RAMP allows connection origination either by a sender or a receiver. A connection originated by issuing a *Connect* request is referred to as an active open, whereas a connection originated by issuing an *Accept* response is referred to as a passive open. The two open modes are needed to control the behavior of the members. For example, in purely passive mode, all receivers wait for a Connect request from the sender first. Upon receiving the request, each receiver responds with an Accept request, thereby joining the group. If one of the receivers is the connection originator, then only the connection originated by this receiver is an active connection. All members must wait then in passive mode until the originator sends the Connect request.

4.4.3 Data Flow

RAMP defines two different data delivery modes in which the sender can operate on, the *Burst* mode and the *Idle* mode. Burst mode minimizes network traffic at the expense of control traffic. This mode is better for smaller receiver groups (nominally, less than one hundred receivers). Idle mode introduces extra messages in the data flow, but minimizes the control flow transfers and is therefore better for larger receiver groups [Koifman/Zabele, 1996]. The sender elects the mode in which it wishes to operate on. Further, the sender is allowed to transition between the two modes as the number of receivers varies.

For both modes, RAMP breaks the data flow into bursts. A burst is defined as a series of messages, where the time between any two messages does not exceed a specific time interval. If the interval between any two consecutive messages is greater than the specified interval, then the first of the two messages defines the last message in the current burst, where the second of the two messages defines the first message in the

subsequent burst. If the sender does not transmit data for a period longer than the specified time interval, then the end of a burst is declared and the sender is required to send one or more Idle messages [Koifman/Zabele, 1996].

4.4.3.1 Burst Mode

In Burst mode, the sender marks the start of the burst by setting the ACK flag. After transmitting one whole burst, the sender must send an idle message indicating that it will remain silent until the beginning of the following burst. All receivers that required reliable delivery must acknowledge the receipt of the message by sending an ACK message to the sender with the sequence number of the packet that had the ACK flag set.

If the sender does not receive an ACK from any of the receivers that required reliable delivery, it assumes that the data to that receiver has been lost, and hence issues a retransmission. If the receiver does not respond after few retransmission attempts, the sender assumes that the receiver has failed and hence it closes the communication channel to that receiver. Figure 28 shows an example of an Active open - Burst mode data flow.

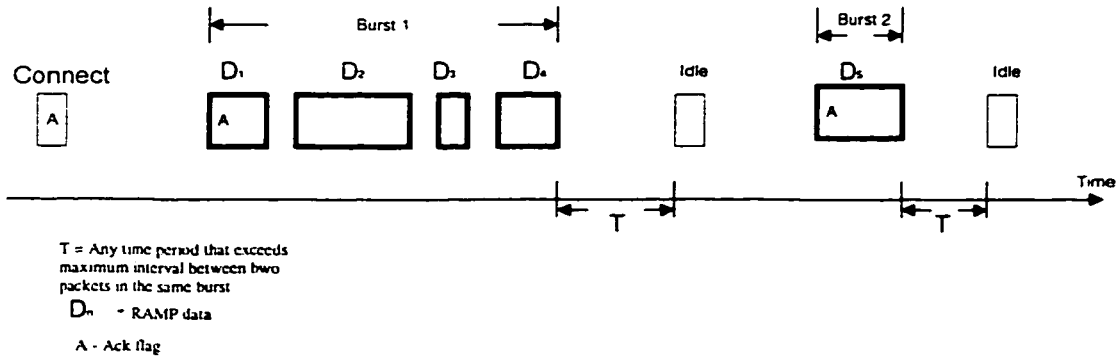

Figure 28: RAMP Active Open, Burst Mode Data Flow.

4.4.3.2 Idle Mode

In Idle mode, a series of Idle messages is sent, each within a fixed time interval, rather than a single Idle message. In addition, the ACK flag, which is used for burst mode to indicate the beginning of a burst, is no longer set in the data packet. The Idle messages are used to inform the receivers that no additional data is available, yet the sender is still alive. If neither data, nor Idle messages are received within the fixed time interval, the receiver assumes a loss of data and hence starts the usual recovery process. If no response is received from the sender after few attempts, the receiver assumes that the sender has failed, hence closes its connection to the sender.

So, in Idle mode, receivers are responsible for identifying and acting on a failed connection, in contrast with Burst mode, whereas the sender is responsible for that. That is the reason why ACK flags are not utilized in Idle mode. By avoiding ACKs, the number of messages that the sender must process is reduced, hence Idle mode is more suitable for large groups. Figure 29 shows an example of an Active open - Idle mode data flow.

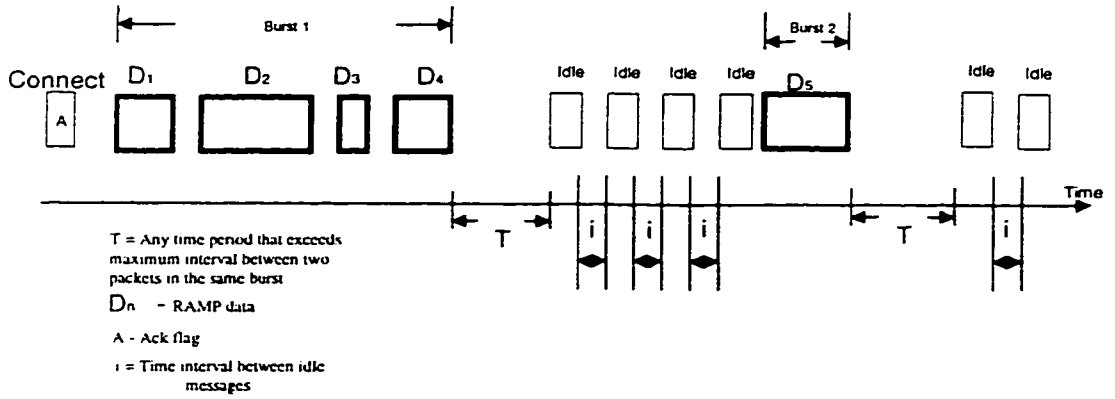

Figure 29: RAMP Active Open, Idle Mode Data Flow.

4.4.4 Connection Style

RAMP can be described as a connection-oriented, reliable stream service as message boundaries are preserved; however, both datagram and stream-style interfaces are supported [Koifman/Zabele, 1996].

All data flows from a sender to the receivers over the data channel using a combination of multicast and unicast, and all control traffic flows from the receivers to the sender over a unicast control channel. With the first implementation of RAMP, initial data transmissions are multicast by the sender to the receiver group, while all retransmissions resulting from lost data are unicast by the sender to the individual receivers.

4.4.5 Error Recovery

RAMP uses immediate, receiver-initiated, NAK-based error notification combined with originator-based unicast retransmission. When a receiver detects a gap in the message sequence, the receiver unicasts a Resend message over the control channel to the sender. The Resend message contains the sequence number of the undelivered message(s) [Koifman/Zabele, 1996]. Depending on the number of retransmission requests received at the sender for a message, the sender decides whether to unicast the retransmission to specific receivers, or to multicast the retransmission to all members.

4.4.6 Flow Control

The flow control approach used in RAMP, calculates a transmission delay factor for each receiver based on the number of NACKs received from that receiver during normal retransmission processing. Since RAMP attempts to provide full reliability, the delay value used by the sender is the greatest of all delay values across the set of receivers. That is, the sender's transmission rate is always adjusted to the slowest receiver.

However, that flow control approach does not prevent the sender from overflowing a receive buffer. This occurs when the sending rate by the sender is faster than the rate in which the receiving application retrieves data from the receiver buffer. In such a situation, the receiver's queue becomes full, however since no segments are lost, the sender does not get an indication to slow down. To avoid this problem, RAMP employs a mechanism in which the receiver intentionally drops segments when the queue becomes greater

than 80% full. This triggers a Resend request by the receiver, providing the sender with the needed throttling information [Koifman/Zabele, 1996].

4.4.7 Unreliable Delivery

RAMP provides two types of unreliable data delivery to support applications where unreliable delivery is acceptable and appropriate [Koifman/Zabele, 1996]. With the first type, the sender and all receivers operate unreliably. That is, the receivers do not send any ACKs or repair requests, and the sender does not process any control traffic from the receivers.

The second type of unreliable delivery involves an unusual model where the sender supports reliable delivery, yet some (or all) of the receivers operate in an unreliable mode. This is appropriate for some applications such as image archiving, where some receivers require a full reliable delivery, while others can tolerate some loss. This type of reliable/unreliable delivery allows a single multicast server to simultaneously support both reliable and unreliable receivers with a single data feed [Koifman/Zabele, 1996].

In RAMP, both the sender and receivers can freely switch between the reliability modes. The sender changes its reliability mode by toggling the reliability flag (bit) in its messages, indicating whether or not receivers are allowed to issue control messages to the sender. Receivers switch between reliable and unreliable modes by simply processing or ignoring lost messages [Koifman/Zabele, 1996].

4.4.8 Experimental Analysis

RAMP performance measurements have been conducted over TASC's internal twisted-pair Ethernet LAN of six Silicon Graphics Indigo (R3000) UNIX workstations [Koifman/Zabele, 1996]. Three types of measurements were performed to test RAMP as a reliable multicast protocol, a unicast protocol, and a concast (multipoint-to-point transmission) protocol.

As a reliable multicast protocol, the effective throughput was ideal, in that it almost matched the theoretical expectations. When packets were intentionally dropped to simulate congested networks, the effective throughput was reduced to 80% and to 66% with a drop rate of 5% and 10% respectively. However, performance continued to increase nearly linearly with the number of receivers [Koifman/Zabele, 1996].

As a reliable unicast protocol, performance was measured as a function of the independent sender-receiver pairs on the same network. The results showed that the aggregate throughput to receivers remained nearly constant when the number of receivers was increased. This continued to be the case even when packets were intentionally dropped. Throughput was reduced to 90% and 74% with a drop rate of 5% and 10% respectively.

As a concast protocol, the throughput rate reduced when the number of senders was increased. Additional reduction occurred with an increase in the drop rate - a reduction to 94% and 92% with drop rate of 5% and 10% respectively. Although the throughput reduction in all cases was only slight, the results made it clear that a more scalable flow control approach is necessary for high volume concast communications with a large number of senders [Koifman/Zabele, 1996].

4.4.9 Protocol Evaluation

4.4.9.1 Strength

The most obvious strength of RAMP lies in its design assumption, where the protocol is to be used over high performance networks. For collaborative interactive applications with matching conditions, RAMP may be the most suitable protocol to use.

The different reliability levels provided by the protocol, from being fully reliable to fully unreliable, makes it equally suitable for many applications with very different reliability requirements.

4.4.9.2 Weakness

The two methods of data flow in RAMP may consume a relatively high amount of bandwidth [Petitt, 1996]. In Burst mode, the overhead is composed of sending an idle message at the end of a burst, and sending an ACK by each receiver at the beginning of each burst (except the first one). This overhead may become significant in cases where data bursts are of small sizes. In Idle mode, the overhead of bandwidth consumption is composed of all the idle messages that the sender needs to transmit should it become silent. This overhead may become significant if the amount of transmitted data is relatively small compared to silent times between bursts.

RAMP assumptions about the high-performance underlying network are very strict. If the network, however, experiences any unexpected problems then the protocol may behave improperly. For example, the flow control method is based on the number of retransmission requests and assuming that more retransmission requests indicate a congestion situation, hence the sender slows down its transmission rate. If the packet losses however, were caused by a poor or malfunctioning link, then the transmission rate should be increased to compensate for the packet loss in that link. Reducing the transmission rate in such a situation will produce undesired effects, especially since the sender assumes a slow receiver and adjusts its transmission rate accordingly, causing all members in the session to be delayed.

4.5 The Reliable Multicast Protocol (RMP)

RMP is a reliable multicast protocol that provides a scalable, totally ordered and atomic multicast service on top of an unreliable multicast datagram service such as IP Multicast. RMP provides a wide range of guarantees, such as unreliable delivery, totally ordered delivery, K-resilient, majority resilient and totally resilient atomic delivery. The RMP design attempts to build a transport layer service that is as comprehensive as possible. Having a robust transport primitive that provides reliable delivery, ordering of messages, and fault-tolerance unburdens the application developer and allows concentration on the development of the application itself [Montgomery, 1994].

4.5.1 RMP Entities

RMP is organized around three entities: RMP Processes, Token Rings, and Token Lists. An RMP Process can be thought of as a member of a process group that is using RMP to provide its transport mechanism. A group of processes, communicating to achieve message ordering, is referred to as a Token Ring. Each process may be a member of multiple token rings. Alternatively, processes may not be a member of any token ring and may communicate with a token ring through the use of a client-server communication model [Montgomery, 1994]. RMP allows hosts that are not multicast capable to participate as members of the ring. Finally, a Token List is a list of members in a token ring. A token list is created by one process, called the Originator. Each process in a list maintains a current version of the token list for protocol references and operations.

4.5.2 Interaction Model – Post Ordering Rotating Tokens

RMP is based on the Post Ordering Rotating Token Interaction Model, in which a token is rotated among group members, and messages are ordered, by the token site – the current member that holds the token – after they have been sent. The model requires all members to maintain a copy of the token list. The token list is updated upon any changes to the members' status; for example when new member joins the ring or when any member leaves. Whenever any member wishes to change the token list, it multicasts a special request, called the List Change Request (TCR). The token site then serializes all the requests, creates a new token list, timestamps and sends it to the members. Thus an ordering of the changes within the global ordering is achieved [Montgomery, 1994].

Figures 30 and 31 illustrate how messages are ordered with the post ordering rotating token approach in two different scenarios [Petitt, 1996]. In both figures, the event order shows the order of events as seen on the network, while the imposed order shows the global timestamps assigned to these events. The scenario in figure 30 is as follows. Initially, site B is the token site. Site A sends the first data message, and assigns a sequence number of 1 to it; that is shown by Data(A, 1). Site C also sends its first data packet, Data(C, 1).

Since the token site, site B, receives Data(A, 1) first, it imposes the order as Data(A, 1), Data(C, 1). Site B then generates an acknowledgment message, which contains three parts: the imposed order, the next token site, and the global timestamp. So an acknowledgment such as ACK((A, 1), (C, 1), C, 1), would indicate that the message from A is to be ordered before the one from C, so even if other sites have received Data(C,1) first, they must reorder their data accordingly. Additionally, it indicates that C is the next token site, and this information has a global timestamp of 1.

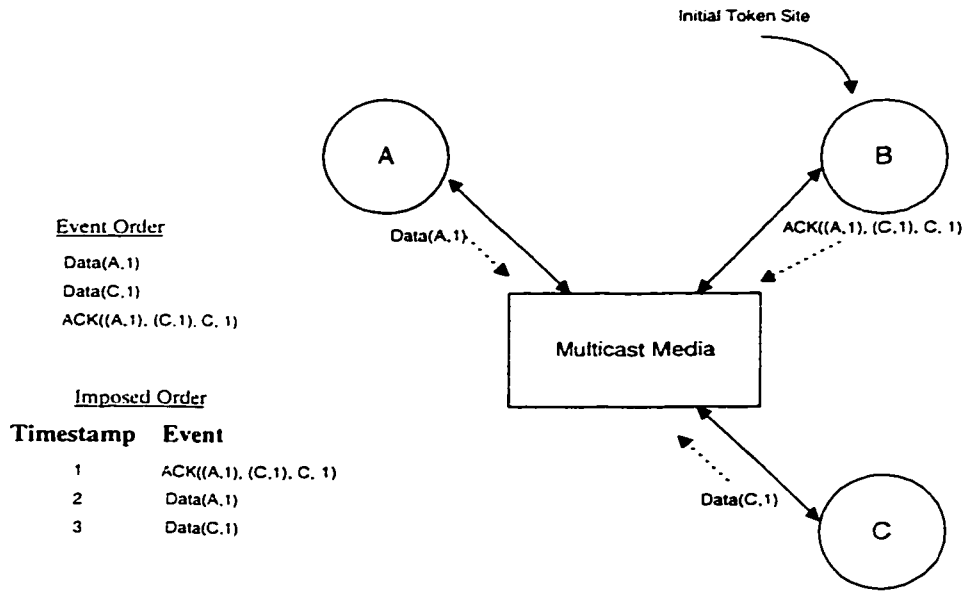

Figure 30: RMP Rotating Token Example

The example in figure 31 actually continues the scenario of figure 30. Now site C is the token site; notice that site C must receive all timestamped messages before it can become the new token site. However, for the period of time when C was the token site, no messages were transmitted. Hence, C transmits an acknowledgment with the first part indicated as NULL, which means no order is to be imposed. The ACK includes site A as the new token site and a global timestamp as 4, which is a continuation from the previous scenario. Now site A is the token site. Site C then wishes to leave the communication, so it multicasts a LCR indicating that. Since this is the second message from site C within the session, the message has 2 as its sequence number. Once site A receives the LCR message from C, it creates a new token list (NL) and

multicasts it to the group. Since this NL message should give the same information that usually an ACK message would provide, the message includes the same three parts of an ACK message. So, a NL((C, 2), B, 5) message is multicast by site A, indicating the imposed order as well as the next token site, site B, and the global timestamp, 5. Similarly B can only accept the token, hence become the new token site, after it has received all timestamped messages [Montgomery, 1994].

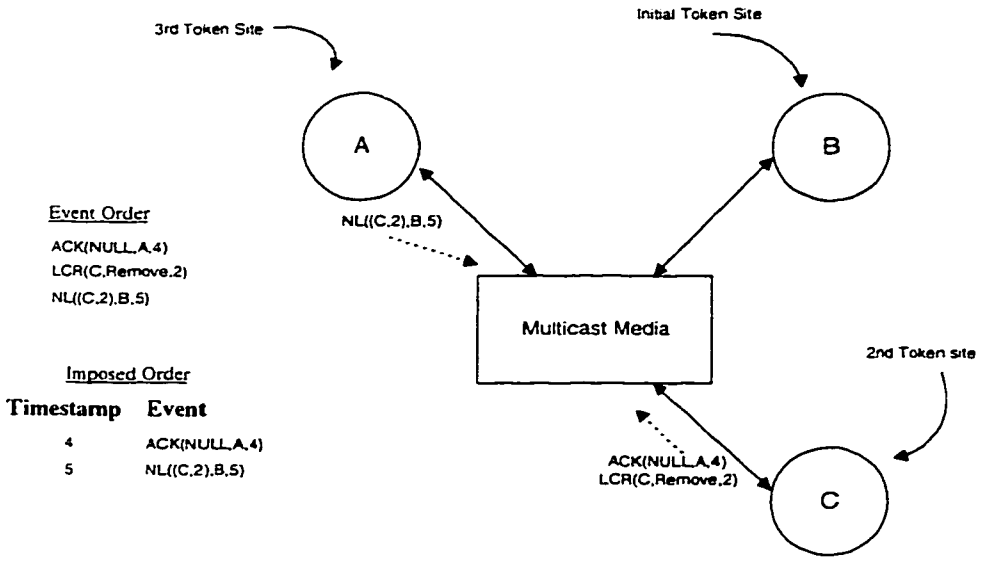

Figure 31: RMP Rotating Token Example (Continuation of example shown in figure 30).

4.5.3 Quality of Service (QoS)

Since not all applications require the same level of atomicity and resiliency, RMP modified the commitment policy of a message to be more flexible. Each message is to be committed depending on its desired QoS. The QoS ranged from unreliable delivery, where the message is committed upon reception, to totally resilient, where the message delivery has the same semantics of the post ordering rotating token described above. Table 5 shows the different QoS levels provided by RMP [Montgomery, 1994], and how the protocol achieves each of them.

QoS Level	Description
Unreliable	Delivery is immediate upon reception. Lost messages are not requested. These messages are not assigned sequence numbers by the sending site.
Reliable	Delivery is immediate upon reception. Lost messages are requested. These messages do receive a sequence number.
Source Ordered	Delivery is after all messages from the same source and with lower sequence numbers have been delivered.
Totally Ordered	Delivery is after messages with lower timestamps have been delivered.
K Resilient	The same as Totally Ordered but with (K-1) passes of the token required as well.
Majority Resilient	The same as K Resilient but with K being equal to (N+1)/2, where N is the size of the ring.
Totally Resilient	The same as K Resilient but with K being equal to N, where N is the size of the ring.

Table 5: RMP QoS Levels

4.5.4 Error Recovery

RMP uses receiver-initiated error recovery, if reliable QoS is required. Errors are detected by checking the sequence numbers. Once a gap is noticed in the sequence number, the receiver multicasts a NACK to the group requesting the missing data. However, to avoid implosion, the receiver waits for a period of time before sending the NACK. The receiver cancels its request only if either another NACK for the same data is heard, or the required repair is received within the delay time; otherwise the receiver multicasts its NACK. Similarly, to avoid repair implosion, the sender/receiver that wishes to send a repair, delays itself for some time before multicasting the repair. The repair is multicast only if no other repair for the same data is heard within the delay time.

4.5.5 Flow and Congestion Control

Flow and congestion control policies used by RMP are designed to be orthogonal to the rest of the protocol. Flow and congestion control policies can be inserted easily into the protocol, and different policies can be used in different environments. As the default, RMP uses a modified sliding window protocol based on the Van Jacobson algorithms used in TCP [Callhan/Montgomery, 1996].

4.5.6 Failure Recovery

RMP implements a Reformation Protocol to be used for failure recovery. The reformation protocol involves two phases. The first phase consists in creating and synchronizing a new token list. The second phase is concerned with committing the list. In the first phase, the site that detects the error, referred to as the Reform Site, polls all other members to detect which of them are still active. Active members inform the reform site with their highest consecutive timestamp and the highest consecutive sequence number they have received from each site in the old token list. Members that have missed packets recover using the usual RMP error recovery procedure, and then update the reform site after they get the needed repairs. For the recovery to be considered successful, a minimum number of members must remain in a partition after a failure. This number is specified in the old token list. If the minimum partition size criteria can not be reached, then the new list is invalid [Montgomery, 1994]. If the recovery is successful, then the reform site multicasts the new list to the group. After each member in the new list has acknowledged receiving the new list, the reform site commits the list and makes itself the token site. This ends the reform protocol recovery operation, and allows normal operation to resume.

4.5.7 Multi-RPC Delivery

Multi-RPC delivery is a facility provided by RMP to allow RMP processes that are not members of a group to send data to a ring and to receive ACKs of successful delivery, as well as responses by a member of the group [Callhan/Montgomery, 1996]. This facility is advantageous since it allows processes to communicate with the ring without incurring the overhead of joining it. In general, these non-member

messages can either be automatically acknowledged, or a single member of the group can be selected to handle a request and reply to it. Additionally, these non-member messages can be delivered with any of the QoS levels available to messages sent by group members.

4.5.8 Experimental Analysis

Experimental tests of RMP have only been conducted over a LAN environment. All tests were run on several Sun Microsystems SPARCstation 5 running SunOS 5.3 operating system on a 10Mbps Ethernet. Aggregate throughput, single sender throughput, and packet latency for different QoS levels were measured. The RMP performance results were compared to the performance of other reliable multicast protocols. While declaring that no fair comparison with other protocols can be made since other experiments have been conducted in completely different environments, Montgomery stated that no other reliable multicast protocol has been able to show as high throughput performance as RMP.

The aggregate throughput is the throughput seen by the application. It is computed by multiplying the total amount of data sent from all senders by the number of destinations. Senders are also counted as destinations. The aggregate throughput was obtained with a totally ordered QoS level, which is actually the worst case for the throughput because of the load on the senders. The tests showed that the aggregate throughput exceeded the Ethernet maximum throughput of about 10^6 Bps. It is not possible for a non-multicast or non-broadcast scheme to break that Ethernet throughput boundary [Montgomery, 1994].

The single sender throughput was calculated as the aggregate throughput divided by number of destinations, where the sender is not counted as a destination. The tests showed that the single sender throughput almost matched the theoretical expected throughput. Additionally, throughput stayed roughly constant as the number of destination increased. Finally the tests showed that higher QoS resulted in higher packet latencies. However, packet latencies remained relatively constant as the number of destinations increased.

4.5.9 Protocol Evaluation

4.5.9.1 Strength

The wide range of quality of services provided by RMP, from unreliable to totally resilient makes the protocol suitable to a diverse number of applications with different requirements. The protocol provides additional flexibility such as the ability for hosts that are not multicast capable to participate as members of the ring, and the ability that members can communicate with the group without being members of the ring, using multi-RPC.

The use of the post ordering rotating token model adds strength to the protocol. For example, rotating the token allows the processing load to be equally distributed among members.

The Reformation Protocol used by RMP to recover from network failure is another strength of RMP. The ability to reconstruct the ring after the network is partitioned makes the protocol very suitable for environments where network partitions are highly expected.

4.5.9.2 Weakness

The scalability provided by protocol is very limited. That is due to the protocol design itself. The protocol must maintain and keep track of the membership information at each member, which may result in a significant overhead if the number of receivers becomes large. Additionally, to provide total ordering, the protocol needs to be connection-oriented, which limits scalability further.

RMP does not specify any concrete methods for flow or congestion control. One problem that RMP faces with flow and congestion control is that the rotating token site introduces a higher overhead per acknowledgment than traditional protocols such as TCP [Callhan/Montgomery, 1996]. The basic sliding

window method, used as the default, restricts the protocol to low loss environments, since high loss rates could severely degrade the protocol performance.

Although the protocol attempts to reduce the number of ACKs, by allowing many messages to be acknowledged at once, the protocol consumption of bandwidth is still high since all messages, ACKs, NACKs, NLs, LCRs and repairs are always multicast to the group.

Finally, the method used to remove a member from the group is a bit strict. The question is how long a member is willing to remain in the group after it has decided to leave. The protocol requires the member that wishes to leave to remain connected until it receives the new token list committing its removal from the ring. Further, the protocol requires the member to continue processing any needed repairs if needed until all the packets that it has transmitted are successfully received by the other members. This slow departure method used by RMP is very unrealistic and impractical.

4.6 The Multicast Transport Protocol (MTP-2)

The first version of MTP-2, MTP, was designed by Armstrong in 1992, and specified in RFC-1301 [Armstrong/Freier, 1992]. MTP is based on a hierarchical structure, and uses a single *master* to achieve rate control, ordering, and atomicity as well as to increase reliability [Ott/Bormann, 1994]. The other members within the communication session are *producers* and *consumers*. Producers are capable of transmitting information, while consumers are only recipients. The collection of processes is called a *web*.
Bormann and Ott designed MTP-2, in 1994 as the multicast transport for a teleconferencing communication facility, the Multipoint Communication Service (MCS). MTP-2 is simply an improved version of MTP, which is more general and more suitable platform for MCS. As examples of these improvements, MTP-2 introduced unicasting and the ability to change the master of the web [Ott/Bormann, 1994].

4.6.1 Data Transmission

The master transmits information in the form of messages, each of which consists of one or more packets. Should any producer wish to transmit data, it must first contact the master requesting a token. For that, the producer unicasts a token request to the master. The master then unicasts a response granting a token. The response also includes a message number. The producer must mark all the packets that it transmits with that message number. The token is implicitly returned to the master with the final packet of the message. Packets within a message have an increasing sequence number; thus a combination of message number and packet number is always unique and can used to identify each packet within the session.

4.6.2 Error Recovery

MTP-2 uses receiver-initiated sender oriented error recovery. Once a receiver detects an error, it should immediately unicast a NACK to the producer. Repairs are multicast to the group. However, MTP-2 does not provide total reliability, so there is no definite guarantee that the requester gets the needed repair. The reason behind that is due to the fashion in which the protocol handles data transfer and retransmission. The data transfer and retransmission in MTP-2 is based on dividing time into heartbeat intervals. After the initial transmission of a packet, consumers have a limited time to request a retransmission of a packet. After that time, producers are no longer obligated to honor NACKs [Bormann/Ott, 1994].

Figure 32 shows an example of MTP-2 transmission and error recovery operations. The empty packet concerns atomicity and is explained in section 4.6.4.

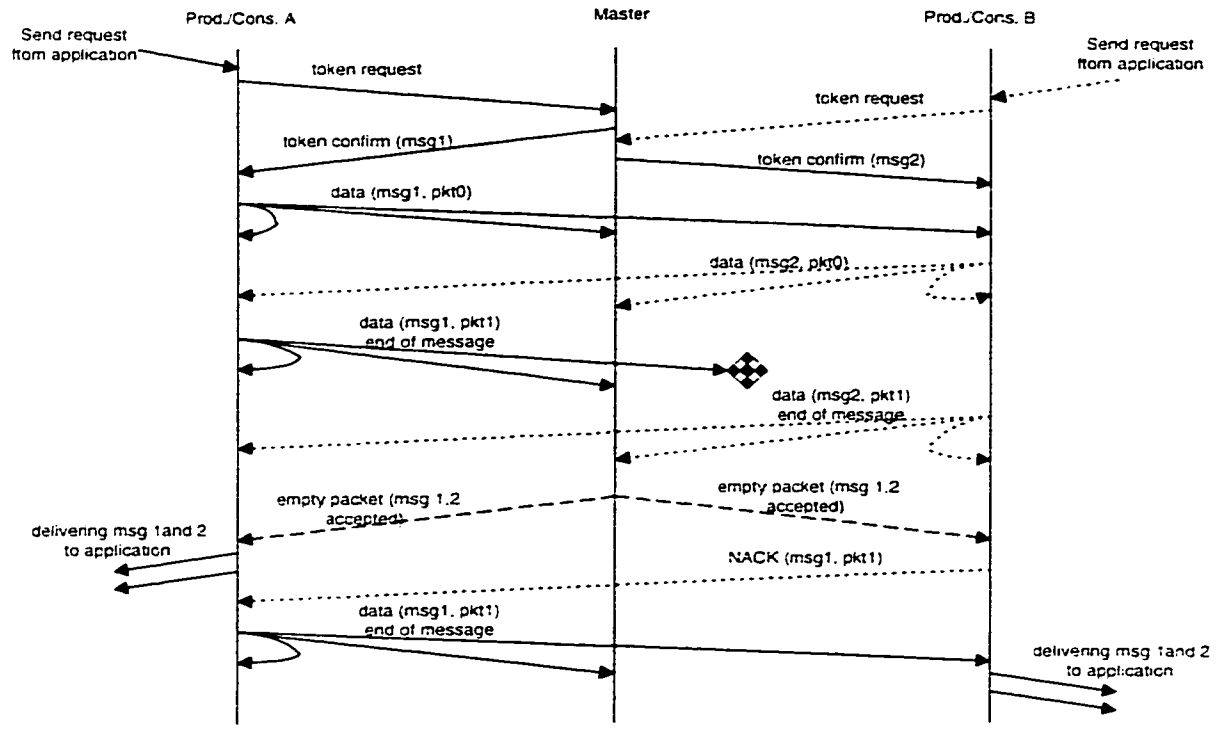

Figure 32: Example of MTP-2 Transmission and Error Recovery Operations.

4.6.3 Ordering

MTP-2 provides global packet ordering. This is achieved by requiring all producers to first get a token from the master sender and attach the message number given by the master along with the token to the transmitted data. Once the packets are delivered to the consumer, it is the consumer's responsibility to deliver the messages in the correct order to its application.

4.6.4 Atomicity

In MTP-2, it is the master's responsibility to provide atomicity. In general, a message may not reach a consumer correctly because either the consumer has failed or the producer has failed. The master maintains a message acceptance record, which assigns a status to the most recent 12 messages in the session. The message status can be pending, accepted or rejected. Pending status indicates that the message

is still in progress. Messages that have been correctly received are marked accepted, while messages that have not been correctly received are marked rejected. The message acceptance record is transmitted in every packet sent by the master. For example, the status is sent to producers that request a token with the token confirmation message. If no producers are active, the master multicasts empty packets including the acceptance record, one per heartbeat. The acknowledgment by the master provides atomicity. A message that was not received correctly by the master is rejected and not delivered to the application.
Since different applications may not require atomicity, MTP-2 allows the group members to disable atomicity, hence messages can be delivered to the application without waiting for the master acceptance.

4.6.5 Flow Control vs. Rate Control

MTP-2 does not provide flow control. Rather, the protocol defines a window, which limits the number of data packets that can be sent per active message in a heartbeat interval by one producer [Bormann/Ott, 1994]. This effectively limits the data rate per message, justifying why the protocol does not need to provide any flow control mechanisms.

4.6.6 Master Loss and Migration

In MTP-2, the master is considered to be a single point of failure. MTP-2 provides a mechanism for master loss recovery. The recovery starts by an attempt to reach the master. A special message is multicast from the members to the master for that. If no response is received from the master, the members elect a new master. A new web is then created, with a new web-id, removing any ambiguity as to whether members still believe to be attached to the failing master [Bormann/Ott, 1994].

Master migration may be necessary if the current host where the master is running is overloaded, or if the network connection to the master is congested. To allow master migration, a procedure similar to the one used for master loss recovery is used [Bormann/Ott, 1994].

4.6.7 Experimental Analysis

Experimental results were mentioned very briefly, with not much significant information, in [Bormann/Ott, 1994]. A few test applications have been written to test the protocol performance. Examples of these applications included a multicast file transfer, a multicast version of UNIX keyboard chat program talk(1) and a multicast audio phone program.

The file transfer program could not compete with TCP, but it did work well at the chosen data rate. The chat program and the telephone program were actually representing low delay and real-time applications. Although MTP-2 is not intended to cover real-time requirements, they held up well [Bormann/Ott, 1994].

4.6.8 Protocol Evaluation

4.6.8.1 Strength

MTP-2 overhead is relatively small. For every message sent, there is two additional packets (token request and token confirm packets) and one round-trip time delay [Bormann/Ott, 1994]. This overhead can actually be eliminated altogether if the communication is one-to-N, by migrating the master to the sender/producer.

The global ordering provided by the protocol is very accurate, since all message transmissions are controled by a single member, the master. Additionally, the protocol adds more flexibility such as the ability to prioritize the token requests and messages.

The ability for the protocol to recover from master loss, or network partition as well as migrating the master if needed, adds strength to the protocol.

4.6.8.2 Weakness

The reliability provided by MTP-2 is weak, since repairs are not guaranteed after a specified amount of time from the original transmission. MTP also relies exclusively on NACKs for error recovery, which limits reliability and requires extreme amounts of buffer space [Callhan/Montgomery, 1996]. This level of reliability may make the protocol unusable for many applications. However, this may not be considered as a severe problem for some applications such as teleconferencing, which the protocol originally was designed for.

The protocol may not be that scalable, since it is utilizes receiver-initiated sender-oriented error recovery. Additionally, since NACKs are unicast to the producer, it is very possible that the producer may end up multicasting duplicates of the same repair if the NACKs are not received around the same time. Finally, having a fixed size, 12, for the message acceptance record introduces less flexibility to the protocol.

4.7 The Local Group based Multicast Protocol (LGMP)

The Local Group based Multicast Protocol (LGMP) supports reliable and semi-reliable transfer of both continuous media and data files [Hofmann, 1996]. LGMP is based on the Local Group Concept (LGC) principle, in which members are organized into a hierarchical structure that is composed of many subgroups, called local groups (LGs). Error recovery and feedback processing are then performed locally within the different groups.

In LGMP, no manual or administrative intervention is necessary to form the local groups. LGMP subgroups are self-organizing and self-adapting, thus improving fault-tolerance and system reliability. Receivers dynamically organize themselves into subgroups, and then automatically select a Group Controller (GC) to coordinate local retransmissions and to process feedback messages [Hofmann, LGMP web site]. The selection of group controllers is not a part of LGMP itself; rather a separate protocol, called the Dynamic

Configuration Protocol (DCP), is used to handle that operation. In general, the selection of group controllers is based on the current state of the network and on the receivers themselves.

LGC local retransmissions and local acknowledgment processing do indeed represent a major part of the principle's strength. With local retransmissions, group controllers are able to perform and coordinate the retransmissions of lost or corrupted data within their groups. This reduces retransmission delays and decreases the processing load on the sender, as well as the load on the network. With local acknowledgment, group members send their control traffic only to their controller. The group controller then sends one control unit to the sender informing it about the status of its group. This avoids implosion at the sender and reduces the number of control units that the sender needs to process. Thus, using the LGC principle as the base of LGMP allows the protocol to be very scalable.

4.7.1 Acknowledgment Scheme

LGMP uses three types of acknowledgments, positive acknowledgments (ACK), negative acknowledgment (NACK), and semi-negative acknowledgments (SNACK). A group controller sends an ACK for a packet to the sender if all members of its local group receive the packet correctly. This allows the sender to release the data unit from its buffer. If none of the members have received the packet correctly, then the group controller sends a NACK to the sender for that packet, which initiates the sender to retransmit the packet. If the packet is received correctly by at least one of the local group members, but not by all members, then the group controller sends a SNACK to the sender. The SNACK will require the sender not to release the data unit from its buffer; however, the sender is not required to retransmit the packet yet. Since local error recovery is to be performed first, failed receivers attempt to get the packet from any member in their local group that has successfully received the packet. If this attempt is successful, and all receivers in the local group have successfully recovered the packet, then the group controller sends an ACK to the sender; otherwise the local error recovery operation is to be repeated. There is a possibility however, that the operation ends up with the group controller sending a NACK to the sender. For example, if the packet was initially received successfully by only one member of the group, then the group controller

sends a SNACK to the sender, and the local recovery operation starts. If, however, the successful receiver dies before transmitting the packet to any of the other members, then the packet can be recovered only from the sender, by sending a NACK. That is why the sender is not allowed to release its buffer upon receiving a SNACK.

4.7.2 Local Error Recovery

LGMP provides two modes of local error recovery, the Load-Sensitive mode and the Delay-Sensitive mode. It is up the application to choose which one of the two modes to use. Additionally, different group controllers can possibly operate in different modes.

4.7.2.1 Load-Sensitive Mode

With load-sensitive mode, group controllers aim to minimize the network load caused by local retransmissions as much as possible. To achieve that, group controllers dynamically determine whether to multicast retransmissions to the whole group, or unicast the transmissions to specific receivers. The decision is taken based on a predefined threshold. If the number of requests for a specific packet exceeds that threshold, then the packet is retransmitted to all group members using multicast; otherwise the packet is unicast to the requesting receivers. However, local retransmissions performed by regular receivers are always multicast to the local group.

4.7.2.2 Delay-Sensitive Mode

With load sensitive mode, group controllers have to delay retransmissions for a specific amount of time in order to collect most/all requests for a packet, and hence get a true estimate of the number of requests for the packet. However, this delay may not be acceptable for many applications with time constraints. To provide proper service to such applications, group controllers can alternatively operate in delay-sensitive mode. In this case, upon receiving a NACK for a packet, group controllers immediately

multicast the retransmission. Further NACKs received for the same packet within a specified time period from the retransmission time are ignored.

In delay-sensitive mode, if a group controller itself has missed a packet, it attempts to get it immediately from any of its local group members, by multicasting a request to the local group. Members of the local group, who have successfully received the packet, delay themselves for a certain amount of time before multicasting a retransmission. A receiver only sends the retransmission if its delay time expires before any other repair(s) for the same packet is seen. This method basically limits the number of retransmissions.

4.7.3 Congestion Control

LGMP provides mechanisms to detect network congestion based on the status reports from each receiver. However, the method of handling a congestion situation is up to the application. For example, the application may slow down the sender if the network experiences a high loss rate. If the receiver set includes some slow receivers, the application may choose to drop out and ignore those receivers. LGMP also provides mechanisms to dynamically create and announce additional IP groups. Each of these groups may transmit at a different rate. Receivers may chose an appropriate rate and join the corresponding IP multicast group [Hofmann, 1997].

4.7.4 The Dynamic Configuration Protocol (DCP)

The placement of group controllers and the assignment of receivers to appropriate local groups are essential for the efficiency of data transfer. Simulations of MBONE scenarios have shown that inappropriate group hierarchies result in inefficient network utilization, in increased transfer delay, and in reduced throughput [Hofmann, 1997]. However, since constructing and managing of logically structured group hierarchies is not a task of data-level protocols, LGMP does not handle such issue itself, rather a separate protocol was designed to handle these issues; that is the Dynamic Configuration Protocol (DCP).

A few important issues have to be examined by DCP when the hierarchy is structured. For example, the processing load should be fairly distributed among the different groups. The method used to create and maintain local groups must be able to handle the failure of any host, especially the group controller. An auto recovery mechanism is necessary to repair the dropped-out group controller. Additionally, since LGMP allows receivers to join an in-progress session, or leave before the session terminates, DCP was designed to allow dynamic creation and reconfiguration of local groups.

Finally, DCP is not restricted to LGMP, but it can interact with any other protocol that requires a logically structured receiver hierarchy [Hofmann, 1997].

4.7.5 Experimental Analysis

Experimental tests were conducted to evaluate the benefits of the Local Group Concept. The evaluation focused on two important measures, the transfer delay, and the network load. The simulation scenario consists of a sender and various receivers connected to a local area network. The sender is linked to the LAN over a wide area network. Additionally, the evaluation examined the impact of group size on average transfer delay and network load. The values obtained for the Local Group Concept were compared to the corresponding results for two common sender-initiated techniques using multicast and unicast retransmission.

The tests showed that LGC significantly reduced the transfer delay compared to the other protocols, especially when the number of receivers was highly increased. Additionally, when the number of receivers in a local group was restricted to a maximum of 50 receivers, LGC produced even a better performance. The high processing load on the sender can explain the significant increase of delay time of the other two protocols.

To evaluate the network load, 50 receivers were added at the sending side of the wide area network. The test compared LGC to the other two protocols, in which one uses unicast retransmissions and the other uses

multicast retransmissions. LGC produced a very low network load compared to both protocols. This is due to the fact that for both unicast and multicast techniques, the ratio of retransmission and total data traffic over the wide area link depends directly on the number of recipients, which, in contrast, has no influence on LGC.

4.7.6 Protocol Evaluation

4.7.6.1 Strength

The most obvious strength of LGMP is its scalability. The hierarchical structure of the protocol along with the local error recovery approach, allow the number of receivers to be highly increased without causing the sender to be imploded.

The two error recovery methods that LGMP provide allow much flexibility, and make the protocol suitable for applications with different retransmission requirements.

LGMP clearly separates transport layer issues from the session layer ones, by having the group management to be handled by a separate protocol, DCP.

4.7.6.2 Weakness

LGMP places some buffer requirements on the receivers. Since releasing the receivers' buffers may be delayed for the purpose of local error recovery, extra memory at receiving sides is required to accommodate that.

LGMP requires a separate multicast address for each local group, which could be a potential weakness if the number of local groups is highly increased.

4.8 The Xpress Transport Protocol (XTP)

The Xpress Transport Protocol (XTP) is a high performance transport protocol designed to meet the needs of distributed, real-time, and multimedia systems in both unicast and multicast environments [Atwood, 1996]. The XTP design makes it very flexible and very suitable for a vast number of applications. The protocol includes explicit support for reliable multicast, yet it provides complete support for unicast communications. Mechanisms for data, flow and error control are provided, but it is up to application to turn any of these mechanisms on or off as desired. Additionally, XTP mechanisms are orthogonal to one another. For example, XTP provides mechanisms for shaping rate control and flow control independently [XTP Forum, 1995]. XTP is also independent of the underlying data delivery service; it only requires the data delivery service to provide framing and delivery of packets from one XTP host to another. Other features provided by XTP include: implicit fast connection setup for virtual circuit paradigm, key-based addressing lookups, parameterized traffic and quality of service negotiation, message priority and scheduling, selective retransmission and acknowledgment and an aligned, 64-bit fixed size frame design [XTP Forum, 1995].

4.8.1 The Communication Model

A XTP *context* is defined as the collection of information comprising the XTP state at an end-system. Each context manages both outgoing and incoming data streams at an XTP end-point. Multiple contexts can be active at one end-point. At receivers, incoming packets must be mapped to a context by consulting a translation map. Senders can take advantage of an optimization, called *full context lookup*, which allows packets to be matched to their context without performing a lookup in the translation database. A context is to be established and initiated before a conversation can start. XTP multicast data flow is simplex, while XTP unicast data flow is duplex. In the case of multicasting, data packets are delivered from a sender to a set of receivers; the opposite direction can carry control traffic. The aggregate of active contexts and the data streams between them is referred to as an *association*.

Since XTP allows multiple active contexts at an end-point/host, the protocol identifies individual contexts by a *key* value. The key value is assigned by the host and is unique within the host. The key is attached to all outgoing packets to identify the sending context. Packets in the return direction carry a return key value, which is the same key with its most significant bit set. This return key is used to uniquely map the packet onto the appropriate context, thus making it easy to reference the information about the context. A key exchange is a procedure by which the initiator of the association is told the key value assigned to the responder context, so that return keys may be used in both directions [Atwood, 1996].

4.8.2 Multicast Group Management

XTP provides a powerful mechanism for group communication, which supports a data transfer service for a one-to-many data flow [XTP Forum, 1995]. Reliable multicasting in XTP requires the sender to have full knowledge of the receiver set. Based on the group information maintained by the sender, the user can manage the multicast group. In the beginning, the user specifies how the initial group of active receivers is compiled and what is the admission criterion. Once the multicast group is established, any receiver may get removed from the active group, or entirely from the receiver set, based also on the information provided by the user.

As a part of XTP usual flexibility, the XTP specification does not specify the form or the content of the group management data structure. The specification, however, recommends the type of information that the transmitter should maintain about the receiver set.

4.8.3 Late Joining Receivers

XTP allows receivers to join an in-progress session. To achieve that, a late joining receiver needs to multicast a JOIN packet to the group. Although the request is multicast to all members of the group, only the sender is allowed to respond to the request. If the sender accepts to admit the receiver, it responds with a JOIN packet, containing the key value for the multicast context and a sequence number of the next data

stream in the session. The receiver can start then receiving new data beginning with the sequence number indicated in the JOIN response packet.

4.8.4 Leaving the Multicast Association

There are three ways for a receiver to depart from the multicast association: voluntary exit, forced exit and silent exit. A receiver voluntarily leaves a multicast group by sending a CNTL packet indicating that it wishes to leave. The receiver then goes into a zombie state for a specified period of time, in which it is required only to respond to packets sent to it by the transmitter via the receiver's unicast host address. As the receiver leaves the group, the transmitter updates its receiver set information.

A multicast receiver can be forced to leave the association. This operation is initiated by the transmitter, which sends a packet to the receiver indicating that it must leave. Once received this packet, the receiver must immediately abandon the multicast association and go quiescent. The transmitter then updates the receiver set information. If the packet from the sender is lost, the receiver may still send control packets to the transmitter. The transmitter can either ignore these packets or send another packet to inform the receiver that it must leave the association.

A receiver may silently be removed from the receiver set. This situation occurs, for example, if the receiver fails. If the receiver fail to respond to status requests, the transmitter starts a synchronizing handshake with the receiver. If the receiver fails to respond, the transmitter assumes that the receiver has failed and hence removes it from the receiver set.

4.8.5 Group Reliability

XTP supports different levels of group reliability. If the user reliability semantics requires that all members of the group remain alive and current, then the detection of a failed receiver dooms the group integrity and results in the termination of the whole association. In this case, the transmitter sends a special packet to all the receivers indicating that the association is to be terminated. After that, the transmitter may

go into a zombie state if required, in which it can only recover for missed/corrupted packets up to the packet that terminated the association.

The user reliability semantics may require only k-reliability. If the user indicates that at least k receivers must be operational for the association to continue, then the transmitter can verify the receiver set against this requirement and terminate the association if the number of receivers falls below the minimum number at any point of time. This level of reliability is concerned only with the number of operational receivers, and not with the identity of those receivers.

Finally XTP supports hybrid reliability requirements by the user. That is, for the association to continue, some specific members must be always present. The transmitter in this case verifies, in a regular basis, the group membership state information against this requirement and terminates the association if any of those specified receivers either fails or exits the association.

4.8.6 Error Control

XTP uses a combination of sender-initiated and receiver-initiated error recovery. Periodically, the transmitter asks the receivers for updates. In normal situations, the receivers just wait until the transmitter request this information before sending any control packets to the sender. However, should a receiver detect an error, it can immediately send a NACK to the sender indicating the error. However, this is provided as an option by the protocol. Control packets indicating lost data contain the highest consecutive sequence number received, and the missing spans of sequence numbers. All repairs are multicast to the group. The retransmission can be either Go-Back-N, or Selective Repeat.

A synchronizing handshake operation is provided by the protocol to assure the transmitter of the state of any receiver. This operation is intended to eliminate spurious decisions that might be caused by network timing anomalies and packet loss [XTP Forum, 1995]. Synchronizing handshake can be initiated by either the sender or the receiver.

Finally XTP provides the flexibility of disabling error control mechanism altogether if needed. This is called no-error mode. However, the sender may still need to ask the receivers for control information to regulate data flow. In no-error mode the receiver needs to include the highest sequence seen in the packet simulating that all packets have been successfully received.

4.8.7 Flow and Rate Control

Flow control operates on end-to-end buffer space. Rate control is a producer/consumer concept that considers processor speed and congestion. XTP provides mechanisms for shaping rate and flow control independently [XTP Forum, 1995].

To regulate flow and error control, the transmitter needs to collect control information from all receivers. The transmitter begins with default or inherited values for flow and rate control. Once the sender requests that a control packet be returned, the receivers include new values for flow and rate control. The sender also periodically updates the round-trip time estimate by observation. In fact, the fashion of handling flow and rate control in XTP multicast is quite similar to the one used for XTP unicast. The packet exchanges carrying flow and rate control parameters between sender and receivers are identical to those between a sender and a single receiver [XTP Forum, 1995].

4.8.8 Experimental Analysis

Unfortunately, although the current release of XTP has been implemented by various organizations and some of these implementations are commercial ones, no performance analysis were reported or made available. Performance analysis of earlier versions of the protocol is available; however, including this analysis is inappropriate, since the current definition of XTP is substantially different from those for which performance analysis studies have been done.

4.8.9 Protocol Evaluation

4.8.9.1 Strength

XTP is perhaps the most comprehensive and mature protocol among the ones evaluated. The protocol provides a very high level of flexibility, which makes it the most suitable protocol for many applications, even with completely different requirements. The protocol is fully reliable, since the transmitter has a complete knowledge of the receiver set. The combination of sender and receiver initiated error recoveries, with the flexibility to turn both of them off if needed, is another strength of the protocol. The protocol support for multicast as well as for unicast is just a prime feature. The complete independence between the protocol's different mechanisms as well as between the different policies can simply be viewed as parts of the great strength that the protocol has.

The protocol has been implemented by various organizations such as the U.S. Navy, and Mentat. Since its first development in 1987, the protocol has been maintained and improved by the XTP forum, which includes some of the most experienced scientists in the field of networking and multicasting. The determination of the XTP forum in continuously enhancing the protocol adds great strength to XTP.

4.8.9.2 Weakness

The most noticeable weakness of XTP is its scalability. The protocol may not be very scalable especially in cases when full reliability is needed. Since the transmitter has to have a complete knowledge of the receiver set, the processing overhead on the transmitter may swamp the transmitter if the number of receivers highly increased. The ACKs and NACKs from receivers can cause implosion at the sender.

Another potential weakness of the protocol is that repairs are always multicast to the group, which may consume unnecessary and possibly significant bandwidth if most of the receivers do not need the repair.

4.9 Too Many Protocols – No Standard!

"Why there are too many protocols? Why there is no standard?" would indeed be very reasonable questions. In fact, the protocols explored in this chapter represent only a subset of the existing reliable multicast protocols. For example, other protocols such as the Adaptive File Distribution Protocol (AFDP) -a protocol originally designed to provide efficient, reliable delivery of large files - and ISIS - a protocol that provides a toolkit for building distributed applications with emphasis on stability and ordering - exist. Other new multicast protocols are expected to be under development at the moment. Unfortunately, the reason behind the existence of many multicast protocols is that none of these protocols was able to fulfil the requirements and needs of all/majority of applications. In depth, the reason behind that is due to the way these protocols were architected. The next subsections explore this subject in detail and identify what exactly these protocols failed to do as a result of their architectural design.

4.9.1 The Architecture and its Consequences

Based on their architecture, the examined existing protocols can be classified as either flat-structured protocols or hierarchically-structured protocols. Flat-structured protocols establish direct connections between the different entities - senders and receivers - of the multicast session. The Scalable Reliable Multicast (SRM) protocol, the Reliable Adaptive Multicast Protocol (RAMP), the Reliable Multicast protocol (RMP), and the Xpress Transport Protocol (XTP) are flat-structured protocols. On the other hand, the Tree-based Multicast Transport Protocol (TMTP), the Reliable Multicast Transport Protocol (RMTP), the Multicast Transport Protocol (MTP-2), and the Local Group based Multicast Protocol (LGMP) are hierarchically-structured protocols.

4.9.1.1 The Flat Structure – The Tradeoffs

Allowing the sender(s) to easily have and maintain a full knowledge of the receiver set could be the most obvious advantage of the flat structure. Since there is a direct connection between the sender and each of the receivers, a full knowledge of the receivers can be maintained at the sender. With the flat

structure, it is also quite easy for the sender to detect the failure of any receiver. The sender may require the receivers to send periodic acknowledgments to it. If no acknowledgments are received from a receiver within a maximum expected amount of time, the sender assumes that the receiver has failed. The sender may then start a handshake process with that receiver to make sure that the receiver has failed. With the flat structure, this handshake process is also very easy to perform.

Another advantage of the flat-structure is its capability to support ordering without adding much complexity to the protocol implementation. In the case of one-to-N multicast, the sender may only need to assign a sequence number to the transmitted packets. The receivers can use this sequence number to correctly order the received packets. In the case of M-to-N multicast, a token can be rotated between the senders to enforce the required ordering. With the flat structure, such token rotation operation can be performed quite simply, since there are direct connections between the senders.

Additionally, with the flat structure, error recovery procedures are relatively easy to implement. Again, this is due to the direct connection between the sender(s) and the receivers. Should a receiver detect an error, it unicasts a NACK to the sender indicating that. The sender also may require the receivers to send their status periodically. So, both the receiver-initiated and the sender-initiated error recovery approaches can be easily used with the flat structure.

Hence, protocols that are designed based on the flat structure can easily support sender knowledge of the receiver set, and ordering. Additionally, the implementation of various parts of these protocols can be relatively simple.

Unfortunately, the disadvantages of the flat structure overwhelm its advantages. The major disadvantage of the flat structure is its scalability limitation. Since each of the receivers communicates directly with the sender, a high number of receivers in a communication session may easily implode the sender.

As a result, protocols that are designed based on the flat structure could ideally be very suitable for those applications where the sender is to have and maintain a full knowledge of the receivers, and when ordering is needed. On the other hand, those protocols are incapable of supporting a large number of receivers. In any event, the following gives a true picture of what the existing flat-structured protocols did and did not support.

The Scalable Reliable Multicast (SRM) protocol did not support ordering altogether, since the Whiteboard application (wb) has no requirements for ordered delivery. In addition, SRM does not take advantage of ability to easily maintain a full knowledge of the receivers at the sender(s). The protocol utilizes the flat receive-oriented receiver-initiated error recovery approach, which causes the protocol to meet a minimal reliability definition. Finally, the protocol is not suitable for applications with a large number of receivers, since it is built upon the flat structure. SRM is considered to be suitable only for the *wb* application, and for other applications with similar specifications and requirement to *wb*.

The Reliable Adaptive Multicast Protocol (RAMP) is fully reliable since it takes a full advantage of the sender knowledge of the receivers. In fact, the protocol is both sender and receiver reliable. Additionally, the protocol guarantees ordered data delivery to all multicast recipients. Unfortunately, the protocol has very strict assumptions about the underlying network, since it assumes a very high-performance network. In addition, the protocol is not scalable since it is designed based on the flat structure. RAMP is useful only for non-scalable applications running over a very high-performance network.

The Reliable Multicast protocol (RMP) provides a reliable totally-ordered delivery of messages. However, the protocol has a main disadvantage, which is its limitation to scale. To provide total ordering, the protocol had to be connection-oriented. Being designed upon the flat structure, the protocol scalability is by default limited. Being connection-oriented, the protocol scalability is limited even further. RMP is not suitable for any scalable applications.

Finally, the Xpress Transport Protocol (XTP) is the last examined protocol that is build upon the flat structure. Although XTP might be considered the most comprehensive and concrete protocol among all existing reliable multicast protocols, the protocol has a very limited scalability. Obviously, this is due to the protocol architecture, which is based on the flat structure. Another disadvantage of XTP is that it supports only one-to-N multicast. Therefore, scalable applications and applications that allow more than one sender to exist within the multicast session can not utilize XTP.

4.9.1.2 The Hierarchical Structure – The Tradeoffs

The most obvious advantage of the hierarchical structure is its capability to scale. Protocols that are built upon this structure are very suitable for applications with a large number of participants. Since large-scale communications shape today, many cooperative applications might only consider the utilization of hierarchically-structured protocols. However, the hierarchical structure has its disadvantages. Since there is no direct connection between the sender and the receivers, maintaining the knowledge of the receiver set at the sender may not be that simple compared to the flat structure. The implementation of other functionalities, such as error recovery and flow control, is also more complex than in the flat structure. Additionally, with the hierarchical structure, some of the multicast features may almost be impossible, or impractical, to support at the same time. Ordering and M-to-N multicast represent examples. In any event, the following examines what the existing hierarchically-structured protocols did and did not support.

The Tree-based Multicast Transport Protocol (TMTP) is suitable for multimedia applications that involve a large number of receivers. The protocol allows members to dynamically join and leave the group. However, RMTP does not assume any knowledge of the receiver set at the sender. At no point within the multicast session, is the sender capable of knowing the identity of the receivers within the session. Additionally, the protocol provides only one-to-N multicast. To conclude, RMTP is useful only for applications with a large number of receivers, where the sender does not need to have any knowledge of the participants.

Applications that require the sender knowledge of the receivers and applications that allow more than one sender within the multicast session can not hence utilize TMTP.

Similar to TMTP, the Reliable Multicast Transport Protocol (RMTP) is a very scalable protocol, but it only provides one-to-N multicast with no support for the sender to maintain the knowledge of the receiver set. As a result, RMTP is only suitable for applications with a single sender and when the sender does not need to have the knowledge of the receivers.

The obvious advantage of the Multicast Transport Protocol (MTP-2) over both TMTP and RMTP is that MTP-2 provides M-to-N multicast, hence it is suitable for applications with more than one sender. In addition, MTP-2 supports global ordering. Unfortunately, MTP-2 has few disadvantages. The protocol is neither that reliable nor that scalable. The protocol does not guarantee full reliability after a specific amount of time from the original transmission. The protocol scalability is not that high compared to other hierarchical-structured protocols since it utilizes receiver-initiated sender-oriented error recovery. To conclude, MTP-2 is not suitable for either applications with full reliability requirements, or for applications with a large number of participants.

Similar to all hierarchically-structured protocols, the Local Group based Multicast Protocol (LGMP) has the advantage of being very scalable. However, the protocol supports only one-to-N multicast. No knowledge of the receivers set is maintained at the sender. As a result, the protocol can only be utilized by scalable applications that do not have more than one sender, as well as do not require the sender to have a full knowledge of the receivers. This limits the utilization of the protocol by many applications.

4.9.2 The Outcome

It is clear that none of the existing protocols was able to provide all the needed multicast features that could be required by a majority of applications. Some of the protocols provide a very high level of reliability, however on the cost of scalability. Some protocols are only suitable for a specific class of

applications. M-to-N multicast is supported only by few protocols. The knowledge of the receiver set is not assumed by most of the protocols. The reason behind all these shortcomings is mainly due to the protocol architecture. A protocol that is built upon a flat architecture can not be scalable due to the implosion problem. A protocol that is designed with a hierarchical architecture can hardly support sender-based reliability, since it is difficult to maintain the knowledge of the receivers set. M-to-N multicast and ordering are very impractical to mix when the number of participants in the communication session is large and when the senders are scattered all over the hierarchy. These limitations and contradictions are directly caused by the architectural design of these protocols. As a result, none of the existing protocols had the success of being suitable to very vast range of applications. This architecture dilemma is in fact the main reason behind the existence of many multicast protocols, rather than a standard one.

The core of this thesis is to provide a solution to this architecture dilemma, which should represent a forward step towards designing a standard reliable multicast protocol. The next chapter, chapter 5, explores our solution in detail.

Chapter 5

A New Architecture for Reliable Multicast Protocols

Although various multicast protocols exist, none of these protocols seems to be capable of providing the functionality required by all, or even a majority of, applications. In fact, some of these protocols were designed merely for a specific application, or a class of applications. In addition, each of the existing protocols either completely ignores some of the multicast requirements, or favors some requirements over others. For example, many protocols do not support M-to-N multicast altogether. Others support M-to-N multicast, however, on the cost of scalability. In fact, the reason that none of the existing protocols is suitable for a broad range of applications is mainly due to the architectural design of these protocols.

Surely, in spite of the fact that applications vary broadly, and the requirements of some applications may even conflict with the requirements of another, a proper architecture of a protocol with a clear separation between policies and mechanisms would enable a protocol to support the majority of applications. Additionally, with a proper architecture, a protocol would be able to provide all of the major multicast requirements at the same time if needed; these are reliability, scalability, ordering, and M-to-N multicast.

In this chapter, we introduce a new architecture for reliable multicast protocols, with which a protocol can efficiently provide a mixture of all the major multicast functionalities, and hence becomes suitable for a vast number of applications. Although our proposed design would allow a protocol to provide all the major multicast requirements at the same time, it is up to the application then to turn any of these features on or off as needed. Moreover, there is no need to build additional new protocols based on the architecture. The architecture can simply be incorporated into many of the existing protocols, thus enhancing these protocols. Surely, new multicast protocols can also be built upon this architecture if wished.

The chapter begins with identifying the problem with all the major reliable multicast protocols. The chapter then examines the multicast features that could be required by most applications/ groups of applications, since these primarily influence the architectural design. The details of our proposed architecture, its various entities and how these entities communicate with each other, will then be presented. A description of how the architecture can be incorporated into some of the existing multicast protocols will then follow. Finally, the chapter concludes with an evaluation of our proposed design and recommendations for multicast protocols.

5.1 Existing Protocols – Architecture, Requirements and Consequences

Distinction can be made between the major existing multicast protocols, based on their design and requirements. The way a protocol is structured directly influences its performance and capabilities. The next subsections examine the different structuring techniques used by the major multicast protocols, along with the requirements and assumptions made by these protocols. The sections also identify the consequences of these design issues.

5.1.1 Sender/Receiver Structure

Each of the existing multicast protocols uses either a flat sender-receiver structure, or a hierarchical structure to connect the sender(s) and the receivers. With the flat sender-receiver structure, there is a connection between the sender(s) and each of the receivers. With the hierarchical structure, receivers are usually grouped into local groups, which compose parts of the hierarchical tree. The hierarchical tree is usually rooted at the sender. Special entities within the tree are designated for controlling the different local groups. These entities could be either members of the receiver sets, or just special agents that are merely responsible for group control and management. The Scalable Reliable Multicast (SRM) protocol, the Reliable Adaptive Multicast Protocol (RAMP), the Reliable Multicast protocol (RMP), and the Xpress Transport Protocol (XTP) use a flat sender-receiver structure. On the other hand, the Tree-based Multicast Transport Protocol (TMTP), the Reliable Multicast Transport Protocol (RMTP), the Multicast

Transport Protocol (MTP-2), and the Local Group based Multicast Protocol (LGMP) use a hierarchical structure to connect the sender(s) and the receivers.

5.1.1.1 Consequences of the Structure

The structure mainly and directly affects the protocol scalability and complexity. Protocols designed with a flat structure have a very limited capability to scale. With a flat structure, whether reliability is achieved using a sender-initiated approach, or a receiver-initiated approach, the number of ACK's or NACK's processed at the sender can easily implode the sender when the receiver set is enlarged, especially when the network is congested or has a high error rate. The flat structure, however, simplifies the protocol design and implementation. Only the set of sender(s) and receivers involved in the session compose the structure; that is, no additional entities are needed for control and management purposes. Additionally, since there is a direct connection between the sender and each of the receivers, it is relatively acceptable to assume that the sender is capable of having a full knowledge of the receiver set, which may be an essential requirement for many applications.

On the other hand, the hierarchical structure allows the protocol to be very scalable. Grouping the receivers into local groups and designating special entities to mange these groups significantly reduces the processing burden on the sender, since local error recovery is attempted first and the number of ACK's or NACK's that is sent to the sender is considerably reduced. Unfortunately, scalability does not come for free. With the hierarchical structure, both design and implementation of the protocol become more complex. Additional entities may need to be added to the hierarchy to control and manage the local groups. Alternatively, some of the receivers can be designated for this group management, but this places a significant burden on these receivers and requires them to have extra capabilities that a receiver simply should not be concerned with. Additionally, since there is no direct connection between the sender and the receivers, as receivers are controlled and served by other entities, it could be difficult for the sender to have a full knowledge of the receiver set. However, a full knowledge could be maintained, but surely on the cost of additional

complexity since this information must be propagated to the sender and periodically be updated by the controlling agents.

Although the utilization of a flat structure may avoid the protocol complexity, it limits the protocol to applications with small scalability requirements. However, since large-scale communications have become common, a protocol incapable of being scalable is indeed reason enough to declare this protocol to be unusable. In other words, scalability is no longer just a simple feature that a protocol may or may not provide; rather it is a core feature that must be supported by any protocol that is designed with the goal of being a standard reliable multicast protocol.

To this end, it is quite clear that the hierarchical structure is favorable over the flat structure. However, other concerns related to the protocol structure rise up. Assuming that the protocol has a hierarchical structure and that the number of receivers that the protocol can support is highly scalable, should the sender(s) have, as well as maintain, the knowledge of each receiver in the receiver set? Would such knowledge practically benefit the application? Would the processing overhead at the sender to maintain such knowledge remain acceptable with a high number of receivers?

5.2 Knowledge of the Receiver Set

How much knowledge the sender should have of the receiver set depends mainly on the application. Many applications, such as news broadcasting, do not require the sender to have any knowledge of the receiver set. In other applications, the sender may have the knowledge of the receivers, yet it can not take advantage of this knowledge in any way, since the receivers may not be allowed to communicate with the sender. Some military communications are examples of such applications. For both these kinds of applications, and similar ones, it is appropriate for the sender not to assume any knowledge of the receivers. The utilization of receiver-initiated error recovery in such cases is considered sufficient to guarantee reliability. The use of a hierarchical structure with these applications allows very high level of scalability, without causing any overhead at the sender.

On the other hand, many other applications require the sender to have a full knowledge of the receiver set. However, in this case, the receiver set is usually small, or the sender needs actually to have the knowledge of a small subset of the receivers. That is, it is very unlikely that the application would require this kind of reliability for a large number of receivers. In other words, although the application may require a very high scalability level, and allows for a large number of receivers, only the knowledge of a specific small set of these receivers might be required at the sender. For example, a multicast application for a political debate would require the interviewer to have full knowledge of the debate members. All questions should be guaranteed to reach these members. Similarly, answers from any member must be delivered to the interviewer and the other members. Clearly, the communication between the interviewer and the members must be based on sender-initiated error recovery, where the sender has full knowledge of the receivers. However, another part of this application would require a large audience to listen to the whole conversation between the members. It would be very unacceptable to require the interviewer to have a full knowledge of the entire audience in the communication session. In such case, even with the use of a hierarchical structure, the sender would be burdened since the processing load on it would still be very high.

Hence, to allow for a very high scalability, yet achieve the required sender-initiated reliability, the sender should only maintain the knowledge of those specific significant members within the session. Additionally, the structure has to be converted from just being a simple hierarchical structure, to a hierarchical structure that places the significant set of receivers in a different way within the hierarchy, as will be explained shortly.

To conclude, a pure flat structure should not be used since it would directly limit the protocol scalability. A hierarchical structure should be used instead, but the sender knowledge of the receivers should be limited only to those significant receivers. Additionally, the set of significant receivers must be placed in the hierarchy in a way that eases the communication between them and the sender and enhances the protocol performance. Finally, this placement method must easily support both ordering and M-to-N multicast.

5.3 Ordering

Ordering is a major requirement for many applications, especially those with multiple senders. However, it is very unlikely that any application would require ordering to take place among a large number of members in a communication session. Rather, ordering would most likely be necessary only between a small and specified set of the members. That is, although the application may be very scalable, ordering is most likely required only for a small set of the senders/receivers within the communication session.

Although some level of ordering can be provided by the application itself rather than by the protocol, this places extra overhead on the application and complicates its design. Hence, it is preferable to provide the needed ordering mechanisms as a part of the communication protocol. Yet, ordering policies should be decided merely by the application. That is, ordering mechanisms provided by the protocol should be completely independent of the ordering policies. As an example, assume the political debate application, where the candidates are to take turns in answering a question. The application may require a interviewer to pass a question to one of the candidates, allow the candidate to answer within a specified amount of time, then pass control back to the interviewer, who in turn passes control to the second candidate, and so on. The ordering mechanisms provided by the protocol should be able to support such a policy. For example, the mechanism should be able to pass a token back and forth from the interviewer to each of the members, and require the interviewer to order the candidate's messages. Alternatively, the application may require a different policy, where the interviewer asks a question, then passes control to the first candidate, who answers the question within an allowed time, then passes control to the second candidate and so on, until control goes back to the interviewer. Similarly, the ordering mechanisms provided by the protocol should be able to support such a policy. For example, a token could be passed in a specified order between the interviewer and the candidates, while requiring each member to place an incremental global sequence number to each message that it sends to guarantee ordering.

To summarize, ordering is a feature that must be provided by any protocol that is designed to be a standard reliable multicast protocol, hence the protocol structure must be capable of supporting this feature. Yet, it is

quite reasonable for the protocol to assume that ordering is to take place in a communication session with a relatively small number of senders/receivers, or between a smaller specified set of the members if the application is very scalable. Finally, the protocol must provide all the needed ordering mechanisms; yet these mechanisms must be completely independent of the ordering policy, which should be determined by the application itself.

5.4 M-to-N Multicast

Having only one sender within the multicast session is not sufficient for many co-operative applications. Many applications require more than one member to be capable of transmitting information. Protocols that allow only one-to-N multicast simply cannot be used by such applications. Hence, for a protocol to be able to serve as many applications as possible, it must be able to support both one-to-N multicast and M-to-N multicast. In any event, the same arguments raised in section 5.3 still hold. Although the application may be very scalable, it is quite unlikely that a huge number of members are to behave as senders. Rather, the number of senders in a communication session is often limited to a small subset of the members. In other words, if the total number of members within a communication session is N, then it should be absolutely sufficient if the protocol supports M-to-N multicast, where M is a small subset of N. Additionally, the support for M-to-N multicast should not cause a high network overhead. For example, a TCP-like style, where M-to-N multicast is provided as M * (1-to-N) multicast sessions should not be used, since its network overhead is too high. In particular, such a technique would require M different multicast addresses; one per sender, and would require each receiver to listen to M multicast addresses as well.

In conclusion, a good design must allow the protocol to support M-to-N Multicast without incurring much overhead neither on the protocol design, nor on the network traffic.

5.5 New Architecture for Reliable Multicast Protocols

As we have mentioned earlier, none of the existing multicast protocols is capable of providing all of the main multicast features. The reason behind that lies mainly on the way these protocols were designed/structured. In the following sections we introduce a new architecture for reliable multicast protocols, with which a protocol can support all of the major multicast features at the same time, which would allow the protocol to be suitable for a very wide range of applications.

5.5.1 Architecture Design

The design distinguishes three sets of entities within a communication session, the set of significant senders/receivers, the set of simple receivers and the set of controlling agents.

The set of significant senders/receivers includes a special member, called the master sender, in addition to those members that are capable of both sending and receiving information. For members in this set, reliability must be fully guaranteed, so sender-initiated error recovery is utilized among those members. Additionally, members within this set compose the M members of M-to-N multicast. Finally, ordering must be guaranteed at all members that belong to that set. The number of members within the significant set is often small.

The set of simple receivers includes all members that are only allowed to receive information. This set can be very scalable. Reliability is achieved within that set using receiver-initiated error recovery. Hence, the identities of those receivers need not to be known to any of the senders in the significant set. Additionally, members of this set are themselves responsible for ordering data sent by the different senders.

The final set is the set of controlling agents. Members of this set are neither senders nor receivers. Rather, they are special entities that are responsible only for controlling different sets of simple receivers, for the purpose of group management and error recovery.

5.5.1.1 The Significant Set

The significant set includes all members that have a special significance to the multicast session. As an example, it would include the interviewer and the candidates in a political debate application, or it would include the set of medical doctors who are running a remote operation. A special member, called the master sender, mainly controls this set. For members in that set, all of the major multicast features are to be guaranteed, if needed. Reliability is guaranteed using sender-initiated approach. The identity of each member within this group must be known to all other members in the set. Each of the members within this group should be capable of transmitting information, hence achieving M-to-N multicast. If required by the application, ordering can be enforced and guaranteed among all members of the set.

Most importantly, all of these features are to be guaranteed without causing either the protocol design to become very complicated, or the network resources to be inefficiently used. Our proposed design achieves that by closely relating those members to each other, as well as achieving all needed communication with the minimal possible network overhead. Although the different members may be physically distant from each other, they must by logically close to each other within the structure. The identity of each member should easily be known to all other members. M-to-N multicast should be achieved using the minimal possible number of multicast addresses, hence minimizing network overhead. The design should also allow error recovery to be efficiently performed.

5.5.1.1.1 Construction of the Significant Set

How the significant set is constructed is indeed a policy that must be left to the protocol itself to decide. Hence, the actual main concern of this section is to make clear what should be achieved when the significant set is constructed. Section 5.5.1.1.1.1 examines this subject. Section 5.5.1.1.1.2 gives an example of how the construction of the significant set can be done to meet the needed requirements. Section 5.5.7.2.1 introduces an existing proposal by Ramasivan [Ramasivan, 2000], which gives an excellent example of how the significant set can be constructed by the Xpress Transport Protocol (XTP).

5.5.1.1.1.1 Significant Set Construction – What should be Met?

Each significant member in the application must belong to the significant set regardless of the physical distance between the members. That is, although the significant members may be scattered within a wide area network, they must logically be close to each other within the significant set.

Each member within the significant set should be capable of transmitting information, hence achieving M-to-N multicast. However, the significant set must be constructed in a way that allows M-to-N multicast to be achieved using the minimal possible number of multicast addresses; that is ideally one.

Additionally, the construction of the significant set should enable each member to easily have and maintain full knowledge of the other members within the set. This would allow error recovery to be achieved using the sender-initiated approach, possibly along with a suppressive NACK receiver-initiated approach, hence achieving full reliability.

The construction of the significant set should also allow ordering to be achieved using any possible ordering policy. This would allow the application to use its desired ordering policy.

Finally, these above features should be provided without adding much complexity to the protocol design and implementation.

5.5.1.1.1.2 How can the Significant Set be Constructed? An Example

To achieve all of the above requirements, our design constructs the significant set as follows. A session manager supplies the identity of each member in the set, and determines which member is to be the master sender. Members may be identified, for example, by their local unicast addresses. In addition, the session manager supplies one multicast address to the members; this is referred to as the significant set multicast address. Once this information is made available to the different members, the needed connections

among the members can be established as follows. Each member initiates connection establishment with the rest by multicasting a FIRST packet to the significant set multicast address: usually the master sender is the first one that initiates the first connection establishment. The FIRST packet indicates that the member is attempting to establish a new connection with the other members. The transmitter of the FIRST packet includes its local unicast address in the packet. In response to the FIRST packet, each of the other members unicasts a JOIN-REQUEST packet to the local address of the member that transmitted the FIRST packet. The JOIN-REQUEST packet includes the local unicast address of the member that transmitted that packet. Upon receiving a JOIN-REQUEST packet from the rest, the member, who initially transmitted the FIRST packet, confirms the connection establishment by unicasting a JOIN-CONFIRM packet to the each of the members from which a JOIN-REQUEST packet was received. The reception of the JOIN-CONFIRM packets completes the connection establishment process.

It should be noticed that a unicast connection is actually established between the sender of the FIRST packet and each of the other significant set members as a result of the connection establishment process. These unicast connections are in fact a side effect of the connection establishment process, and not a requirement of the significant set construction method.

Figures 33 through 35 illustrate an example for a significant set of six members {A, B, C, D, E, F}, where the master sender, A, establishes the needed connections with the rest of the members. Figures 36 through 37 illustrate the similar scenario when member E establishes the needed connections with the rest. Surely, the unicast connection between any two members is only established if it does not already exist. For example, according the two shown scenarios in figures 33 through 38, a unicast connection will exist between A and E once E receives the JOIN-CONFIRM packet from A. As a result, no more unicast connections will be established when A returns the JOIN-REQUEST packet to E. A simply returns the packet using the already existing unicast connection between E and itself.

A final picture of the significant set once all the significant members establish the needed connections is shown in figure 39.

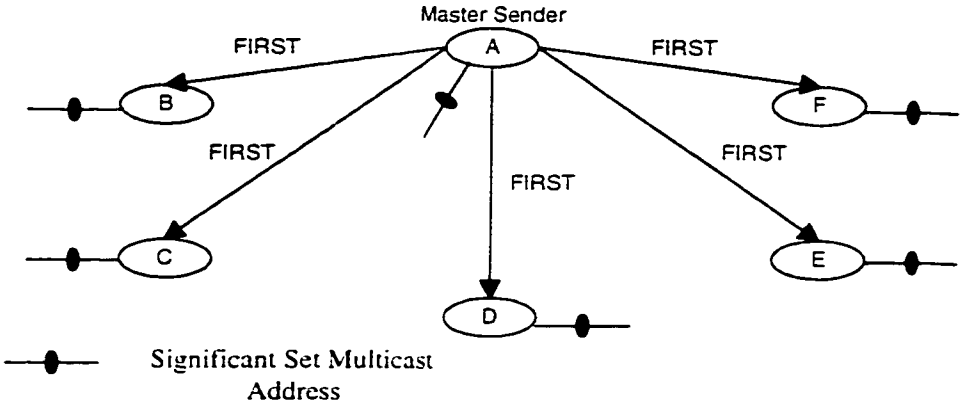

Figure 33: FIRST Packet from the Master Sender to Significant Members

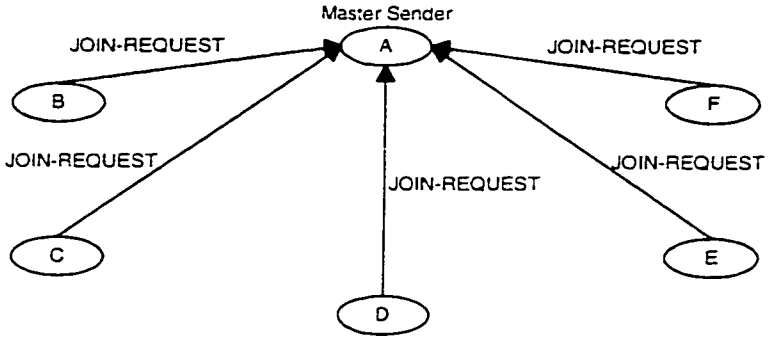

Figure 34: JOIN-REQUEST from Significant Members to Master Sender

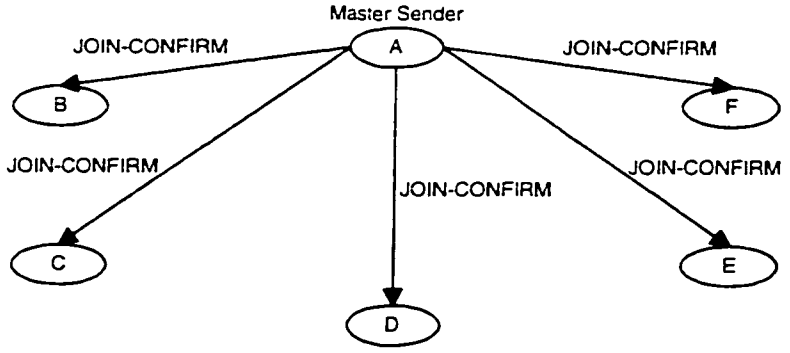

Figure 35: JOIN-CONFIRM from Master Sender to Significant Members

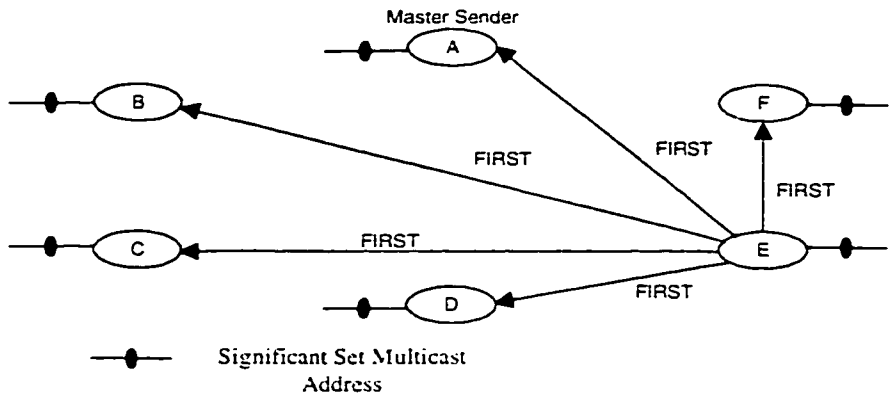

Figure 36: FIRST Packet from a Significant Member to the Rest

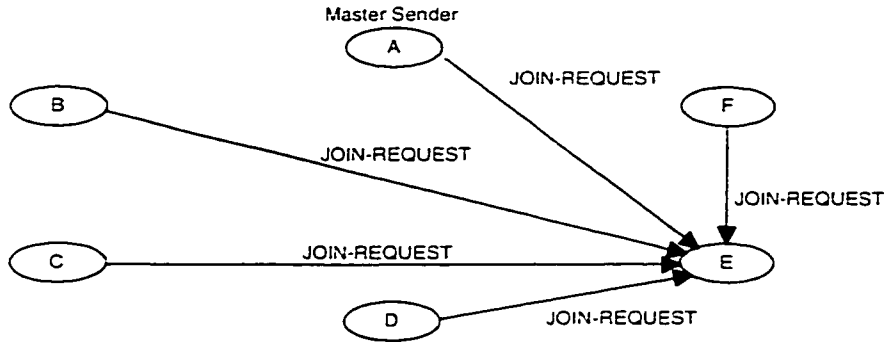

Figure 37: JOIN-REQUEST from Significant Members to a Significant Member

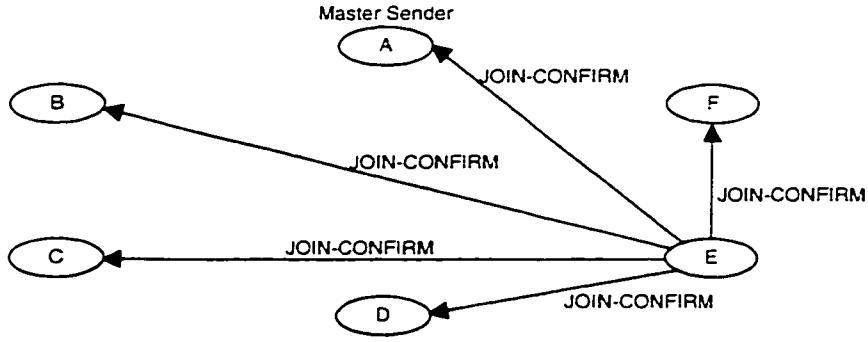

Figure 38: JOIN-CONFIRM from Significant Members to a Significant Member

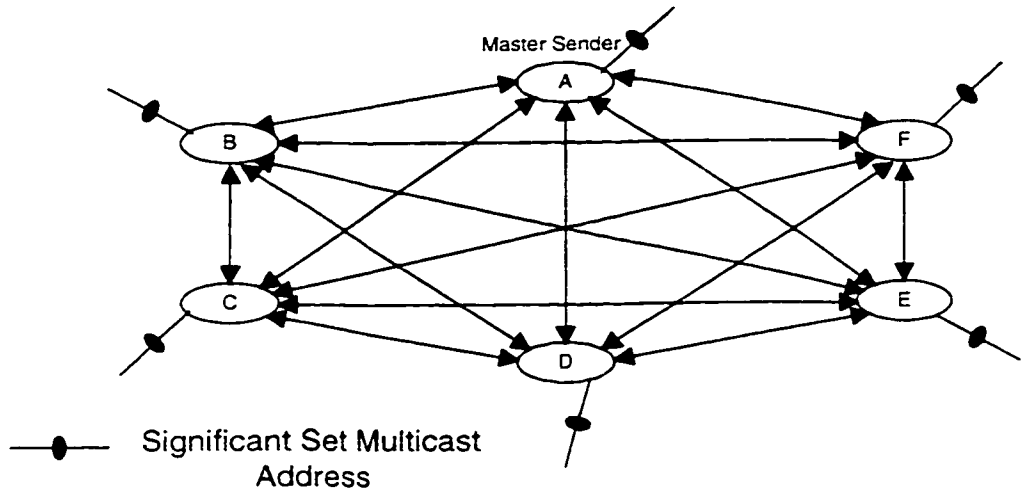

Figure 39: The Significant Set after Connection Establishment between Members

5.5.1.2 The Simple Receivers Set

Unlike the significant set where the number of members is relatively small, the simple receivers set is very scalable. This set includes those members who are only allowed to receive information. As in the political debate example, those members represent the audience. They are only allowed to receive information sent by any of the candidates (members of the significant set). Reliability is guaranteed within this set using receiver-initiated error recovery. That is, it is merely the member responsibility to guarantee reliability. The identities of that set's members are unimportant to the significant set members. Hence, there is no need for any of the senders to either keep or maintain the knowledge about any members of that set. This significantly reduces the overhead on the senders, as well as the complexity of protocol implementation.

5.5.1.2.1 Construction of the Simple Receivers Set

To provide as high scalability as needed, members of the local groups are organized into local groups within a hierarchical structure. The different branches of the hierarchical tree are rooted either at the master sender, or at any of the significant set members. Each local group is controlled by either a

controlling agent, or a significant receiver. Agents are neither senders nor receivers; rather they are special entities that are capable of sending and receiving information only for purpose of group management and error recovery. There is a single multicast address for each local group; this is referred to as the local group multicast address.

To establish the needed connections within a local group, the controlling agent (CA) performs a similar operation to the one used for establishing connections between members of the significant set; see section 5.5.1.1.1.2. The controlling agent multicasts a FIRST packet to its local group multicast address. In response, each member unicasts a JOIN-REQUEST packet to the agent. The agent then confirms the establishment of the connection by unicasting a JOIN-CONFIRM packet to each of the local group members. As a result of the operation, a unicast connection is established between each local group member and the controlling agent of that group. Figures 40 through 42 illustrate these operations for a local group of four members {X, Y, Z, W}.

A complete picture of a local group at this point is illustrated in figure 43. How these unicast and multicast addresses are to be utilized will be explained when exploring error recovery.

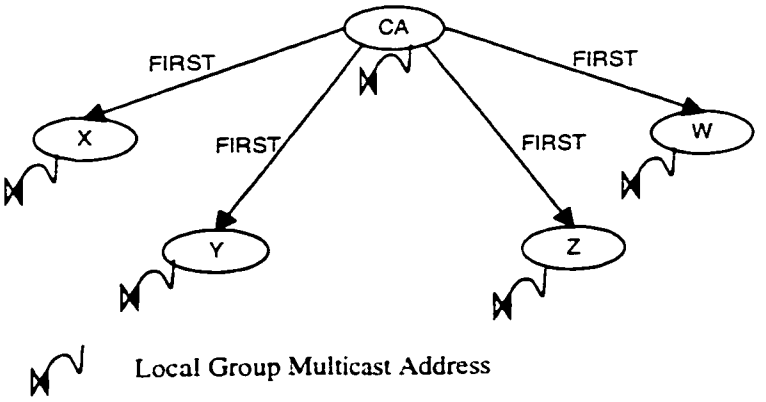

Figure 40: FIRST Packet from Controlling Agent (CA) to Local Group Members

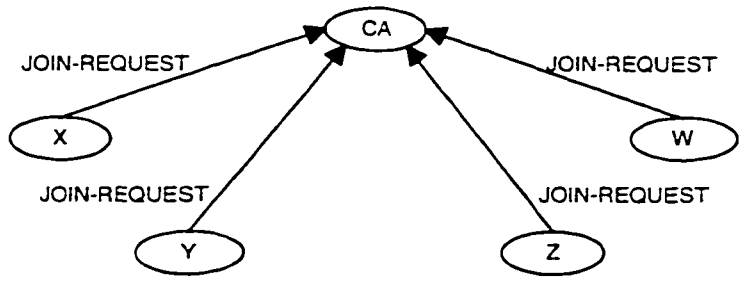

Figure 41: JOIN-REQUEST from Local Group Members to Controlling Agent

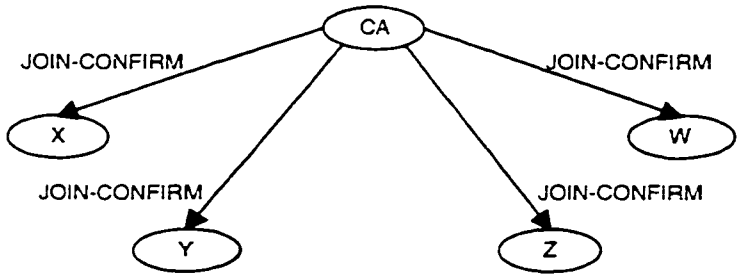

Figure 42: JOIN-CONFIRM from Controlling Agent to Local Group Members

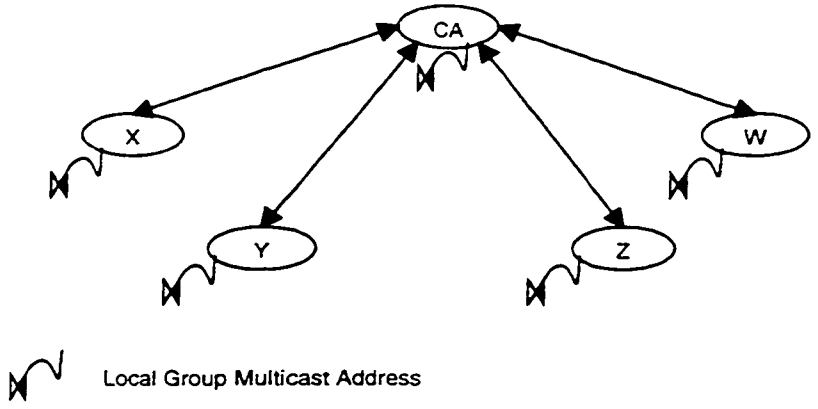

Figure 43: Local Group after Establishment of all Needed Connections

5.5.1.3 The Controlling Agents Set

Although the responsibility of controlling a local group could be placed on any of the members within the communication session, this would require each member to have special additional functionality to handle that. To avoid placing any unnecessary requirements on the members, special agents are placed in the hierarchy to handle such controlling functionality. Agents are neither senders nor receivers, but special network entities that are capable of both sending and receiving data for the purpose of group management and error recovery only. These different agents compose the controlling agent set. Beside those agents, only members of the significant set are required to have the needed controlling functionality. In general, this places very little controlling requirements, since the number of members that is required to have such additional requirements, that is the number of significant receivers, is often small. Although this may slightly burden the significant members, it would increase the protocol's error recovery efficiency since it eliminates one request per each of the tree branches, as will be explained when error recovery is discussed.

Since all controlling operations are handled by either a controlling agent or a significant member, receivers are not required to have any controlling capabilities. However, if a receiver has such capabilities, then it is allowed to behave as both a receiver as well as a group controller.

In conclusion, the set of controlling agents consists mainly of special entities that are solely responsible for controlling operations. Since the significant members are required to have controlling capabilities and to share in the controlling operations, they are considered a part of the controlling set as well. Finally, since receivers with the needed controlling capabilities are allowed to behave as controllers, these receivers also compose a part of the controlling set.

5.5.2 Hierarchy Construction

Assigning the different members to their groups is a session manager responsibility. The different groups are connected to each other though members of the controlling set. Each group member listens and sends to one multicast address, the multicast address of its local group. Hence, each controller listens, and

sends to, two multicast addresses, one for the higher-level group that it is connected to, and one for the group that it controls. A simple receiver may be allowed to transmit to its local group multicast address, but only for the purpose of error recovery, as will be explained in the error recovery section. There is a unicast connection between each of the significant set members and between each simple receiver and its group controller. These unicast connections are actually a side effect of the connections' establishment process as explained in sections 5.5.1.1.1.2 and 5.5.1.2.1.

In addition to these unicast and multicast addresses, there is one special multicast address, called the global multicast address. Each member and controlling agent within the hierarchy listens to this multicast address. Only the significant set members are allowed to transmit data to that address. All new packets transmitted by any of the senders are sent to that address. All receivers and control agents hence listen to that global multicast address.

So in total, each non-controller significant member or simple receiver listens to two multicast addresses, the global multicast address and its local group multicast address. Each controlling agent/member listens to a total of three multicast addresses, the global multicast address, its higher-level group multicast address, and the multicast address for the group that it controls. Figure 44 gives a whole complete picture of the hierarchy. The figure shows a significant set of six members and a few local groups. As shown in the figure, significant members control some of the local groups, while other groups are controlled by controlling agents.

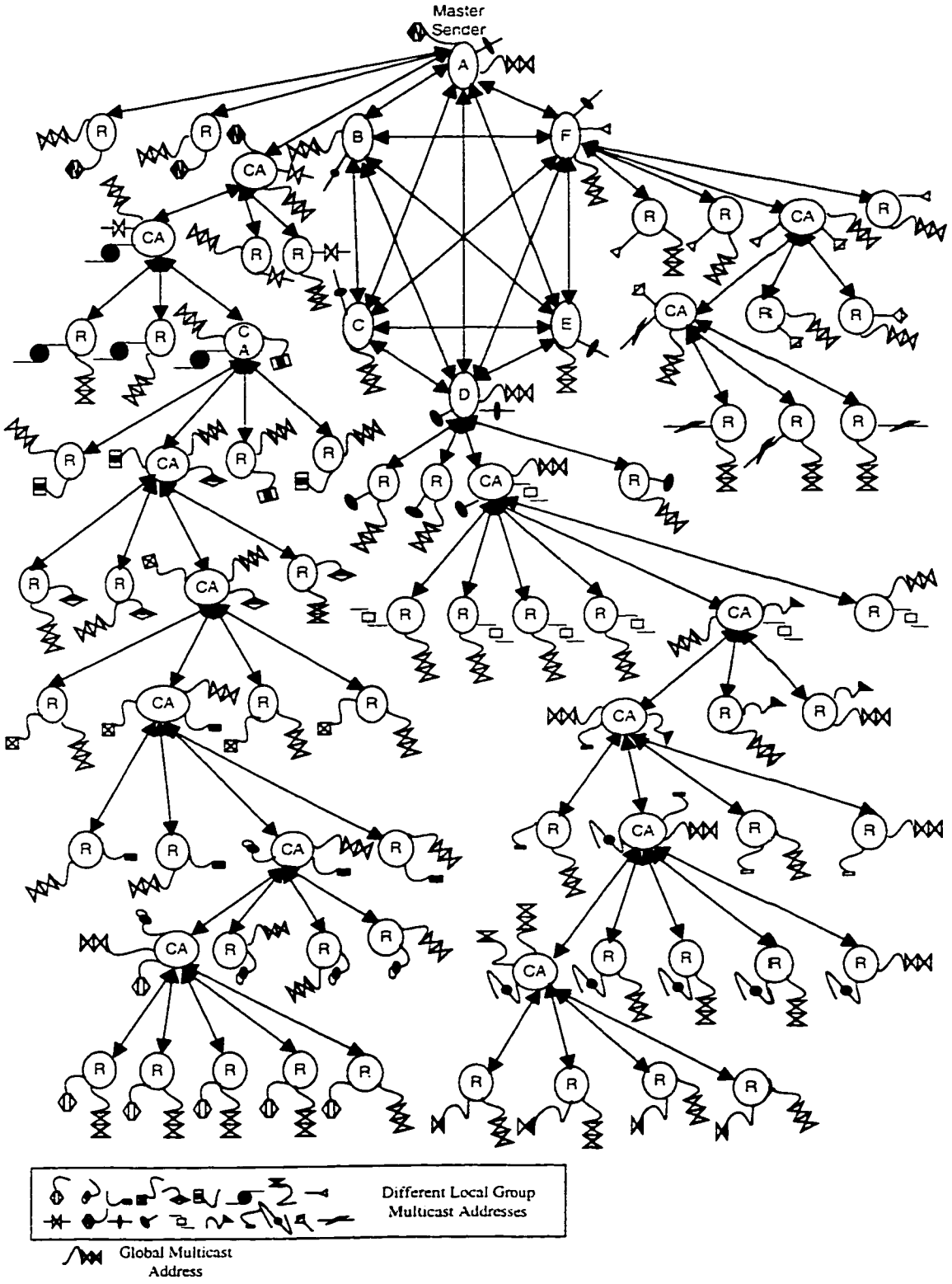

Figure 44: New Architecture for Reliable Multicast Protocols

5.5.3 Data Flow

How data flows and is managed depends mainly on whether the multicast session is a one-to-N session, or it is an M-to-N session.

5.5.3.1 One-to-N Data Flow

In one-to-N data flow mode, only the master sender is capable of transmitting information. Each transmitted packet must include the identity of the master sender and a sequence number assigned by the master sender. This information is needed for error recovery as will be explained shortly. The sender identity can be indicated using its unicast address or a special identification key if the protocol supports that. In the case of one-to-N data flow, the set of significant members may be empty, except for the master sender, or it may contain a number of receivers. Significant members in the later case are not allowed to transmit data to the global multicast address. They are allowed only to transmit information to the significant set multicast address for the purpose of error recovery.

All new transmissions sent by the master sender are multicast using the global multicast address. Each entity within the communication session can receive the packets sent by the master sender by listening to the global multicast address.

5.5.3.2 M-to-N Data Flow

In the case of M-to-N data flow, all of the significant set members are capable of sending data. However it is only the master sender who is capable of starting the session, by multicasting a special packet to the global multicast address.

Once the session has started, any significant member can start sending data. If ordering is enforced by the application, then the transmission is controlled by the ordering policy. All data transmissions are multicast to the global multicast address. Each transmitted packet must include the identity of the sender and a

sequence number local to that sender. This information is needed for error recovery. The sender identity can be indicated by its unicast address or a special identification key if the protocol supports that. All transmissions are simply multicast to the global multicast address. All entities within the session can receive all packets sent by any of the senders by listening to the global multicast address.

5.5.4 Error Recovery

Two different techniques are used for error recovery within the structure. The technique used depends mainly on the identity of the member that missed the information. In the context of error recovery, the member can either be a significant member or any other entity within the session. In fact, the approach used for error recovery within the significant set differs from the one used for error recovery within the local groups.

5.5.4.1 Error Recovery within the Significant Set

Error recovery within the significant set uses a mixture of sender-initiated and receiver-initiated approaches. Since there is a unicast connection between each pair of the significant members, it is quite reasonable to assume that each member has a sufficient knowledge of the rest of the members within the set. The sender uses the global multicast address to transmit any new data. Every specific time period, the sender inquires about the receiving status from the rest of the members, by multicasting a request for status to the significant set multicast address. Hence, only the significant set members see these status requests. Each member sends its receiving status to the requester sender using the unicast connection between itself and the sender. Using these responses from the different members, the sender can detect if any member has missed any packet(s). If packet loss is detected, then depending on the number of significant receivers that missed a packet, the sender may either unicast the missed packet to those specific members that have missed it, or alternatively it may multicast the packet to the significant set multicast address.

In addition to the above sender-initiated error recovery, a receiver-initiated approach is used at each receiver to detect any packet loss. Should a receiver notice a gap on the received packet stream by any

sender, it can either unicast a NACK to that sender, or multicast a NACK to the significant set multicast address. It should however be noticed that a NACK suppression is needed in both cases. The receiver must delay itself for a period of time between sending the NACK. There is a danger in sending an immediate NACK, since many networks reorder packets. NACK suppression in this case would allow the receiver some additional time to either receive the packet if it was reordered, or to be surer that the packet was actually lost.

Which method is to be used is determined by the application. For example, if the network between the significant members is highly loaded, then the application may decide that all NACKs are only unicast to the sender to reduce the network load. If NACKs are unicast to the sender, then it is up to the sender whether to unicast the retransmission to each of the members that requested it, or to multicast the retransmission to the significant set multicast address. The decision depends mainly on the number of members that missed the packet. If this number exceeds a specific threshold, then the sender simply multicasts the packet to whole set, since this is more efficient.

On the other hand, if the application decides that NACKs are to be multicast to the whole significant set, then NACK suppression is utilized. That is, once a receiver detects a gap on the received packet stream, it delays itself for a specified, or possibly random, period of time before multicasting the NACK to the set. The receiver only multicasts the NACK if no other NACKs are sent to the significant set multicast address within the delay time, hence eliminating NACK duplications. If this method is used for NACK requests, then the sender must as well multicast the retransmission to the significant set multicast address, so the packet is received by all members that missed it, but suppressed their NACK requests.

It should be clear that the design is capable of supporting both policies. It is up to the application then to determine which one to use.

5.5.4.2 Error Recovery for Simple Receivers

Error recovery for simple receivers is performed using the receiver-initiated approach. Within the local groups, local error recovery is always attempted first. Once a simple receiver, or a controlling agent, notices a gap in the received packet steam from any sender, it informs its local controller of the loss situation by sending a NACK request for the lost packet. A receiver can use one of two methods to inform the controller of a packet loss. A receiver may unicast the NACK request to its controller using the unicast connection between itself and the controller, or it may multicast the NACK request to its local group multicast address. The application determines which method is to be used, depending on the application requirements or the network status.

Unicasting the NACK requests to the controller has the advantage of reducing the processing overhead at other members. However, it may cause implosion at the controller. In reality, this may not be a problem, since the number of members in a local group is often small enough not to cause significant implosion. If this method is used then once a member detects a packet loss it unicasts its request to its local controller. If the local controller has the requested packet, then depending on the number of request that it has received for that packet, it may simply unicast the packet to each of the members that requested it, or multicast the packet to the local group multicast address. However, if the controller does not have the requested packet, then it attempts to get the packet from any of the group members that did not request the packet. That is, the controller assumes first that the packet can be retrieved locally, and hence attempts a local recovery.

However, there is no real guarantee that any of those members who did not send a NACK have successfully received the packet. For example, there is a chance that those members have not detected the packet loss yet, or they detected the loss and sent a NACK for it, but the NACK itself was lost. In any event, the controller attempts first to get the packet from those members that it thinks may have successfully received the packet. If any of those members has successfully received the packet, then it immediately unicasts the packet to the controller, which in turn decides whether to unicast the packet to the requesters or to multicast it to the local group address.

If the attempt to recover the packet locally fails, then the controller assumes that none of its local group members has successfully received the packet, and hence it escalates the request by sending a NACK for the packet to its higher-level controller. This escalation process is performed in a similar manner. That is, in this case, the controller unicasts the NACK to its higher-level controller. Similarly, the higher-level controller attempts first to recover the packet locally. If this attempt is unsuccessful, then NACK request is escalated further. The NACK escalation continues until either the packet is found at any member in the way up the hierarchy, or the request reaches the sender, which can then issue a retransmission.

Alternatively, a receiver may inform its local controller as well as all of its local group members of a packet loss by multicasting a NACK request to its local group multicast address. In this case, a method for NACK suppression is used. That is, once a receiver detects a packet loss, it delays itself for a specific, or random, period of time before sending its NACK request. If no other requests is seen in the local net within the delay time, then the receiver multicasts its request; otherwise, it cancels sending the NACK request. If any member within the local group has successfully received the packet, then that member should multicast the packet using the local group multicast address. However, to avoid multicasting duplications of the same packet, a suppression method similar to the one used for NACK suppression is used. That is, members that have the packets should delay themselves for a period of time before multicasting the packet to the group. If at any point within the delay time, the packet is seen in the local net then all members who have not multicast the packet yet must cancel their transmissions.

In any event, if no responses have been made to recover for the lost packet within a specific time limit, then the controller assumes that none of the group members has successfully received the packet, and hence escalates a NACK to its higher local group. The escalation operation is done in a similar fashion to the one used within the local group. That is, the controller multicasts a NACK to its higher-level local group in an attempt to recover for the packet loss at this level in the hierarchy. If this attempt also fails, then the higher-level group controller escalates the NACK request further. The escalation process continues until either the

packet is found at any member, or until the request finally reaches the original sender which can then retransmit the packet.

Whether NACKs are to be multicast to the local group or unicast to the controller, local error recovery is always attempted first. This has the advantage of significantly reducing the number of request at the senders as well as allowing for a faster recovery. This is a key point in the structure to allow for a very high level of scalability. Additionally, it is totally up to the application to determine which policy to use. The structure is fully capable of providing the needed mechanisms to support both policies.

5.5.5 Late-joining Receivers/Senders

New members are allowed to join an ongoing session. However, the policy and method of handling these late-joining members depends on the identity of the member. To be specific, that distinction is based on whether the joining member is one of the significant members or whether it is just a simple receiver.

5.5.5.1 Late-joining Significant Members

Indeed, the application requirements completely control whether the multicast session is allowed to start without any of the significant members, or not. In most cases, the application may abort the whole multicast session if any of the significant receivers is absent. For example, in a medical application where a group of doctors must be present for the application to proceed, the absence of one of those members may be sufficient to abort the whole session. In such applications, the support of late joining significant members is not of any significance. However, in other applications, the session may go on without one or more members being present. In this case, the application may require only a subset of the significant members to be present for the session to proceed. In such a case, protocol support for late-joining significant members is needed. Thus, it is absolutely up to the application to define whether late-joining significant receivers should be allowed or not. The protocol however, should be flexible enough to support this operation.

If the application allows a session to start without the presence of a significant receiver, and allows the receiver to join the session late, the receiver can join the application by sending a request indicating that it wishes to join to the master sender. The master sender is the only one who can accept or reject the join request from the late member. However, the master sender may need to consult the rest of the significant members first before permitting the late member to join. The master sender can perform this consultation dynamically by multicasting an informative message to the significant set members indicating the request from the late member to join. Each member should then reply to the master sender with a message indicating whether it accepts or reject that join request from the late member. However, this dynamic consultation may be time consuming and may result in a conflict situation if some members accept and others reject to allow the late member to join. Such situation may lead to a further negotiation between the master sender and the members that reject the request, or alternatively the master sender may be allowed to make its own decision to whether accept or reject the request. To avoid this time consumption and confusion altogether, the decision of whether to accept or reject any specific late member, should it attempt to join, can be taken at the beginning of the session just after the significant set is constructed or alternatively it can be taken based on a predetermined session policy. In this case, the master sender only needs to check the pre-taken decision to decide whether to accept or reject the join request from the late member. In any event, it should be noticed that this decision taking process is absolutely a session layer issue and not a transport layer one.

If the master sender decides to allow the late member to join the significant set, the master sender multicasts an informative message to the significant set members indicating that this late member was granted permission to join. The master sender then replies to the late member indicating that its request to join is accepted. Once the late member gets an acceptance message to join, it starts to establish the needed connections with the rest of the significant set members. The member establishes these connections by multicasting a FIRST packet to the significant multicast address. Each of the members should then respond with a JOIN-REQUEST packet to the late-joining member. The member then confirms the connection by sending a CONNECT-CONFIRM packet to each of the members. Once these steps are performed, the late-

joining receiver can start to listen to both the significant set multicast address and the global multicast address. The application may however, restrict that late joining receiver from transmitting any information before it is updated with the information that it has missed.

Depending on the session/application requirements, the new receiver may start receiving packets only from the moment it joins the application, or it may have to catch up with the rest of the receivers. In the first case, the receiver will not attempt to get any of the older packets that were previously transmitted. As a result, no action is needed either by the late joining member, or by any of the senders.

However, the session/application may require the new receiver to recover from being late and get all/subset of the old transmitted packets. In this case, there are two possible techniques to allow the new receiver to catch up with the rest of the members: Limited Window Transmission, and Full Recovery. The two techniques are explored in sections 5.5.5.3 and 5.5.5.4.

5.5.5.2 Late-joining Simple Receivers

Since most applications allow simple receivers to join an ongoing multicast session, the protocol should be able to support such an operation. The late joining receiver must first get assigned to one of the local groups; this is a responsibility of the session manager. Once the receiver is updated with the local group information, it attempts to establish a connection with the group controller. To establish such a connection, the receiver first sends a JOIN-REQUEST packet to the controller indicating that it needs to establish a unicast connection with the controller. The controller then responds with a JOIN-CONFIRM packet indicating its acceptance to establish that connection. Once these steps are successfully performed, the late-joining receiver is actually a part of the local group of that controller.

Depending on the application requirements, the receiver may then be able to recover all missed packets, a subset of the missed packets, or possibly none of them. If the application does not allow a late receiver to recover any of the missed packets, then no action is needed either by the controller, or by the receiver. In

this case, the receiver simply starts receiving packets from the moment it is connected. If however, the application allows late joining receivers to recover all/subset of the missed packets, then one of the Limited Window Transmission or Full Recovery techniques is to be used to allow such recovery.

5.5.5.3 Limited Window Transmission

With this technique, the new receiver can only get a subset of all the packets that were transmitted before it joins. Using the combination of the sender key/identification number and the sequence number on the new arriving packets, the new receiver can learn what packets it missed. The receiver then attempts to recover all missed packets by sending a unicast request to its group controller requesting all the packets that it missed. (If the late-joining receiver is a significant member, then the recovery request is sent to the master sender). If the multicast session is an M-to-N session, then this request would indicate all missed packets from each of the senders.

With the limited window transmission technique, the group controller / the master sender can only provide a subset of the old transmitted packets. There is a predefined window at each controller that indicates how many packets it can store at a time. The window contains for example, the latest K packets that were received by this controller. If the multicast session is a one-to-N session, then the latest K packets transmitted by the sender are stored at the controller. If the session is an M-to-N one, then the controller stores a combination of the latest K packets transmitted by the different senders. Upon receiving the request from the late joining receiver, the controller can determine whether it has all the requested packets or not. If the controller has all the packets, then it unicasts them to the new receiver. If not, then the controller unicasts a control packet to the receiver, indicating which packets it can deliver. It is then a session/application layer responsibility to either accept to continue without receiving all the old transmitted packets, or abort the operation and leave the session. If the application/session accepts to continue with the limited provided recovery, then the controller unicasts the entire K stored packets to the receiver. The receiver can then start receiving new packets as they arrive.

5.5.5.4 Full Recovery

With the Full Recovery technique, receivers can join an ongoing session at any time and still recover all the data that was sent before they join. This feature has additional requirements. All the senders and the controllers need to buffer all data that were sent during the session. However, this requirement could be very strict especially if the session is a long-lived one. To reduce this buffering requirement on the different hosts, two modifications can be made. The first modification would require only significant set members to store the entire session. Once a receiver joins an ongoing session, it unicasts a recovery request to its controller (that is the master sender if the late-joining receiver is a significant member). Depending on how late the receiver joined the session, the controller may or may not have all the transmitted packets (the master sender however should have all the packets). If the controller has all the needed packets, it unicasts them to the receiver; this would successfully terminate the recovery process. However, if the controller does not have all the old transmitted packets, then it escalates the recovery request up the hierarchy until either one of the controllers can provide the needed packets, or the request reaches the significant senders, which are capable of retransmitting all the old transmitted packets. These retransmissions are unicast down the hierarchy through the different controllers until they reach the late joining receiver.

The second modification would implement a two-level storage space for the significant senders and the controllers. In this case, the most recent packets are stored in memory, while the rest of the packets are stored on disk. If a late joining receiver issues a recovery request, and all of old transmitted packets are in memory, then the controller simply forwards the packets to the requester. If not, then the controller needs to retrieve the old packets from the disk, and then forward them to the receiver.

Although this approach reduces the storage requirement on the significant receivers and the controllers, it unfortunately has an obvious problem, which is the I/O overhead involved with it.

Finally, it is possible to combine the two methods together. That is, require only specific set of the significant senders or the controllers to store the whole data, while allowing these entities to have a two-level storage.

5.5.6 Ordering

Ordering is a core requirement for many applications. However, for practical consideration, ordering should merely be enforced within the significant set since all senders belong to that set. As mentioned before, the ordering policy should be determined by the application. The ordering mechanisms provided by the protocol should totally be independent of the applications policies.

As a matter of fact, it should be possible for any protocol that is built upon our architecture to provide a proper mechanism to support any ordering policy. This is basically due to the way the significant set members are connected to each other.

As an example, if the ordering policy requires a token to move back and forth between some controlling significant members and the rest, the protocol can implement a mechanism that moves this token through the unicast connections between each of the members and that controlling member. Should the application policy be changed to require a token to be passed in a cycle between all/subset of the significant members, the protocol should also be able to support such a policy by passing the token in the unicast connections between those members. If the application does not utilize a token to enforce ordering, but uses some other approach such as a time stamp or a global sequence number, the protocol should also be able to easily provide a supportive mechanism for this policy. For example, senders may send to the global multicast address as usual, but messages get ordered according to the order in which they have arrived at a specific significant member.

To conclude, it should be possible for any protocol that is built upon the architecture to easily provide a proper ordering mechanism to support any ordering policy desired by an application.

5.5.7 Mapping to Existing Protocols

One main advantage of our architecture is that it can be incorporated into many of the existing multicast protocols. In fact, those protocols that already use a hierarchical structure should be able to easily adopt the architecture. Significant changes may be needed to incorporate the architecture to multicast protocols that utilize a flat structure, however these changes are not actually caused by the architectural requirements, rather they result from converting the protocol structure from a flat structure to a hierarchical one in general.

5.5.7.1 Architecture Incorporation into Hierarchically-Structured Protocols

The architecture can be incorporated into the Reliable Multicast Transport Protocol (RMTP) by creating a special local region that includes the significant members. The identity of a significant member can be made available to the master sender and to the rest of the group members by the session manager, which is heavily used by RMTP in any case. Having this knowledge, the unicast connections between the significant set members should easily be established. The Designated Receivers (DR) in RMTP can handle all local region management, which is done by the Controlling Agents (CA) in our architecture. Since there is no need for either the master sender or any of the significant members to have any knowledge of the identity of all simple receivers within the session, no modifications are required to RMTP concerning this subject as the protocol already assumes no knowledge of the receivers.

The error recovery method used by RMTP can still be used as is for simple receivers' error recovery if the multicast session is a one-to-N session. Some modifications are needed to that method if the session is an M-to-N. In this case a simple receiver may need to send a different bitmap for each sender, or possibly the sent bitmap may be modified to indicate the receiving status from each of the senders. Since the significant set is just introduced to RMTP, an error recovery technique for this set is to be implemented by the protocol. This technique can simply be similar to the one that we proposed.

Since RMTP allows late joining receivers, and utilizes two-level caching at the DR's, no significant changes are expected for these subjects. In addition, congestion control can be performed in the same way, by tracking the number of retransmissions by the receivers and using this number as an indication for possible network congestion. Although, this may not be the best approach to detect real network congestion, as described in section 4.3.8.2, our architecture places no requirements on RMTP to change such an approach.

The architecture can similarly be adopted into the Local Group based Multicast Protocol (LGMP). A special local group that includes the significant members is to be added to the LGMP tree. Organizing the significant members into one local group, as well as the selection of the controlling agents, or LGMP group controllers, can be performed and controlled using the Dynamic Configuration protocol (DCP).

Since LGMP already utilizes local error recovery for the receivers, no change is required for the simple receivers' error recovery. Both load-sensitive and delay-sensitive modes can still be used as usual. If the multicast session is one-to-N, then the three acknowledgment schemes (ACK, NACK, and SNACK) used by LGMP can be utilized without any changes. However, if the session is an M-to-N, then the group controllers must be able to acknowledge the different senders with the receiving status of their groups. This can be performed by sending a separate ACK, NACK or SNACK to each of the senders, or by possibly sending one acknowledgment with a general receiving status to all the senders. Regardless of the enforced policy by the application, or the intended mechanism by the protocol, the architecture should easily be able to support the implementation needed to achieve such requirements.

Since the significant set is an add-on to LGMP, the protocol must implement a proper error recovery for that set. The knowledge of the significant members of each other can be easily made available using DCP.

Finally, congestion control can still be performed as it is currently. Based on status reports created by the receivers, LGMP estimates whether, or not, there is network congestion. How a congestion situation is handled can still be left to the application to decide, as it is currently done by LGMP.

5.5.7.2 Architecture Incorporation into Flat Structured Protocols

The architecture can also be incorporated into reliable multicast protocols that use a flat structure, however significant changes might be required in this case. These changes are in fact due to the architecture conversion from a flat one to a hierarchical architecture in general, and not due to the utilization of the architecture. A protocol conversion from a flat structure to a hierarchical one would require building and connecting the different local groups, as well as handling all operations related to these groups, such as error recovery and flow control. As mentioned, this may result in significant changes to the protocol design and implementation, but it is the only way to allow these protocols to become scalable.

For the Reliable Multicast Protocol (RMP), a significant part of its design and implementation can directly be used for the significant set portion of our proposed architecture. RMP already provides M-to-N multicast, where each sender within the session has a full knowledge of the rest. Current RMP members do represent the significant set members of our architecture.

The exact ordering method used by RMP to guarantee packet ordering can still be used as is, since the architecture is capable of supporting it as described in section 5.5.6. In addition, all the QoS levels allowed by RMP, such as unreliable delivery, source ordered delivery and totally resilient, can still be provided as is when utilizing the architecture.

RMP receiver-initiated error recovery can as well be used as is within the significant set, using the significant set multicast address and the NACK suppression to avoid implosion. Flow and congestion control policies utilized by RMP may not also be affected, since they are already designed to be orthogonal to the protocol; so the same modified version of TCP sliding widow can still be used.

So in general, although a serious change may be needed to change RMP structure to a hierarchical structure, a significant part of the protocol design and implementation can still be utilized without much change to enable the protocol to adopt our architecture.

The same above argument holds for the Xpress Transport Protocol (XTP). A significant change to the protocol design and implementation might be needed to convert the protocol from a flat architecture to a hierarchical one. However, a large part of the design and implementation can still be used as is, or with minor modifications, when incorporating our architecture to the protocol. To be specific, the construction of the significant set as well as the communication between its members is expected to utilize much of the current protocol implementation. The knowledge of the significant set members to each other should easily be achieved since XTP already requires the sender to have a full knowledge of the receivers. The bi-directional unicast connections between the significant members as well as between the simple receivers and the controlling agent can be established by sending an XTP FIRST packet with the sender's local key, and with the RCLOSE bit cleared to allow data in the return path. In reply to a FIRST packet, a member should return a JCNTL request packet with its local key and with the sender's local return key. Finally, the sender should respond to the JCNTL request packet with a JCNTL response packet with the receiver local key as well as the receiver exchange key; that is used by the sender to uniquely identify the receiver. Section 5.5.7.2.1 introduces an existing proposal by Ramasivan [Ramasivan, 2000] that details these operations.

So, the current XTP multicast session is somewhat related to the significant set in our architecture. However, the significant set mainly differs from a current XTP multicast session in that it provides M-to-N multicast, which is not part of the default design of XTP.

To conclude, it is quite possible to incorporate the architecture to XTP, and still utilize much of the current implementation. Although a major change may be needed to convert the protocol architecture to a hierarchical one, the outcome of performing these changes is indeed worth the effort. XTP is perhaps the most mature and comprehensive reliable multicast protocol among all the well-known protocols. Having XTP to provide an efficient M-to-N multicast as well being very scalable could possibly turn XTP to become that future standard reliable multicast protocol.

5.5.7.2.1 Multi-sender Communication for XTP

Ramasivan [Ramasivan, 2000] presented a new proposal for multi-sender communication for XTP. Indeed, Ramasivan's proposal represents an excellent method for constructing the significant set for the Xpress Transport Protocol (XTP). In particular, the proposal enables XTP to support M-to-N multicast while using the minimal possible multicast addresses.

In the XTP multicast specification, setting the RCLOSE bit in the XTP header disallows data flow from the multicast receivers to the sender. Ramasivan's proposal, in contrast, clears the RCLOSE bit in the FIRST packet, thereby allowing data on the reverse path of the full duplex channel between the sender and each of the receivers [Ramasivan, 2000].

Ramasivan showed how M-to-N multicast is achieved using an example of an XTP association of four members {T, U, V and W}, where T is the association master. The association master is the process at a given host that first transmits XTP FIRST packets for a given M-to-N group. It is also responsible for distributing transmit tokens and coordinating the communication between the entities [Ramasivan, 2000]. The sender makes use of the local key for the multicast group K_{r_g} for multicast data transmission. The processes at U, V and W each have a separate listening contexts, which wait for the arrival of the FIRST from the association master context at host T [Ramasivan, 2000].

When a host master receives the FIRST packet, it performs a full context lookup. Unlike XTP one-to-N multicast where a full context lookup will fail if the address is in use, the full context lookup does not fail in the M-to-N multicast case, as more than one sender may exist in a group. Figures 45 through 47 show the example presented by Ramasivan for the key exchange operations performed by the association master, T, when it wishes to transmit. Figure 48 shows the data flow at the Transport Layer after these key exchange operations take place.

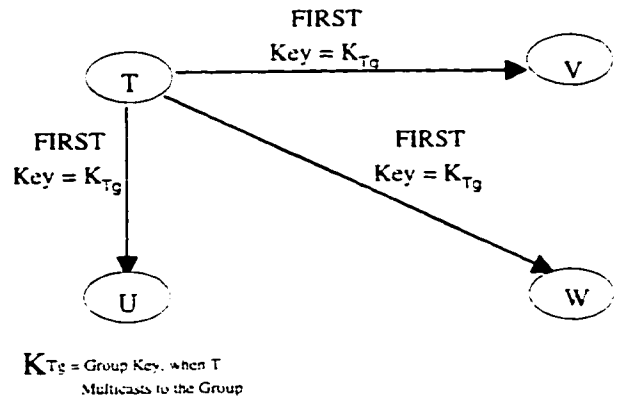

Figure 45: FIRST packet from the Association Master to the group

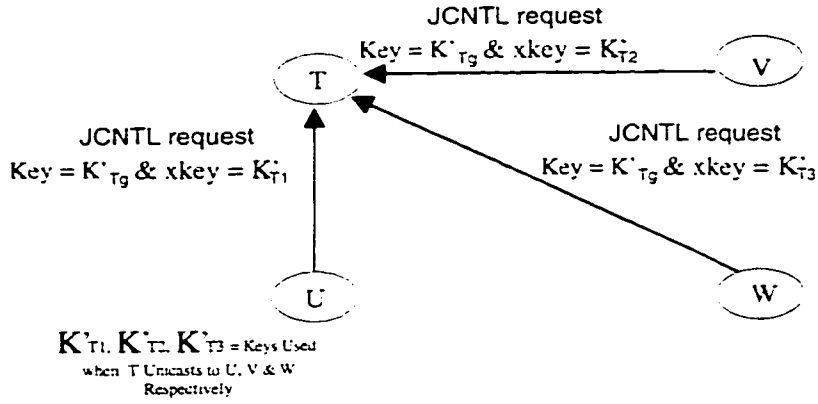

Figure 46: JCNTL request Packet from the Group Members to the Association Master

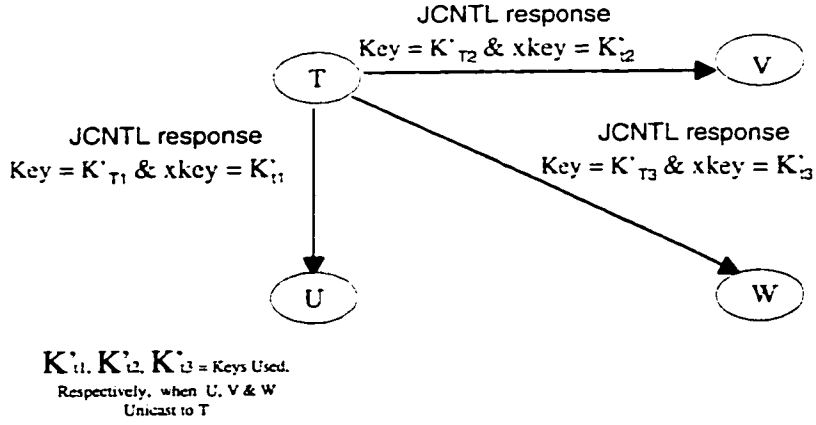

Figure 47: JCNTL response Packet from the Association Master to the Group

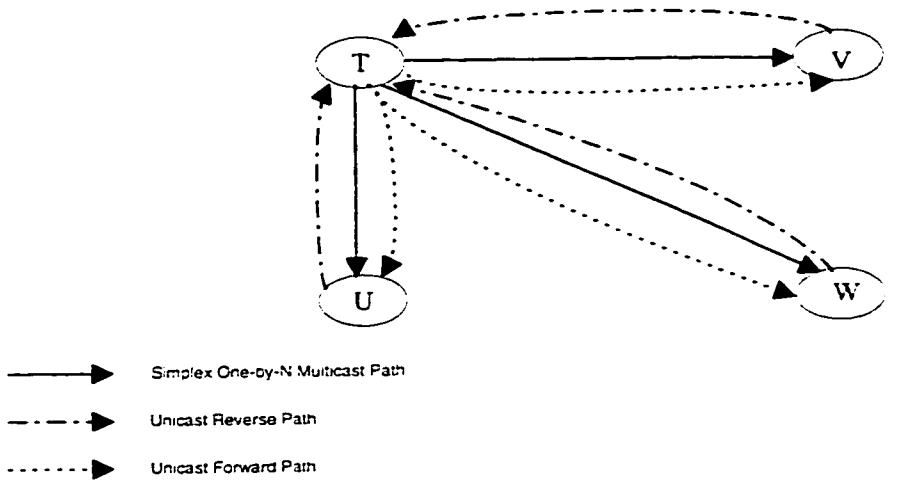

Figure 48: Data-Flow at the Transport layer when Process at T transmits

Each other sender in the association then initiates a similar key exchange operation to the one initiated by T.

Once these key exchange operations are completed, each of T, U, V and W will be capable of multicasting data to the entire group using the group multicast address. In addition, and as a side effect of the key exchange operations, there will be one unicast connection between each of the members.

In addition to this M-to-N communication facility, Ramasivan proposed a design change, where a separate communication layer, referred to as the Enhanced Support Communication Layer (ECSL), is built on top of the modified M-to-N XTP multicast. The ECSL provides a set of primitives that the application can use to participate in a multi-party communication [Ramasivan, 2000]. ECSL will incorporate a number of building blocks listed in the taxonomy and will allow XTP to be efficiently used for the purpose of M-to-N communication [Ramasivan, 2000].

Figure 49 shows the architecture of the XTP M-to-N communication model presented by Ramasivan. For simplicity, the figure assumes that each host has only process that is interested in being a part of the multi-party conversation.

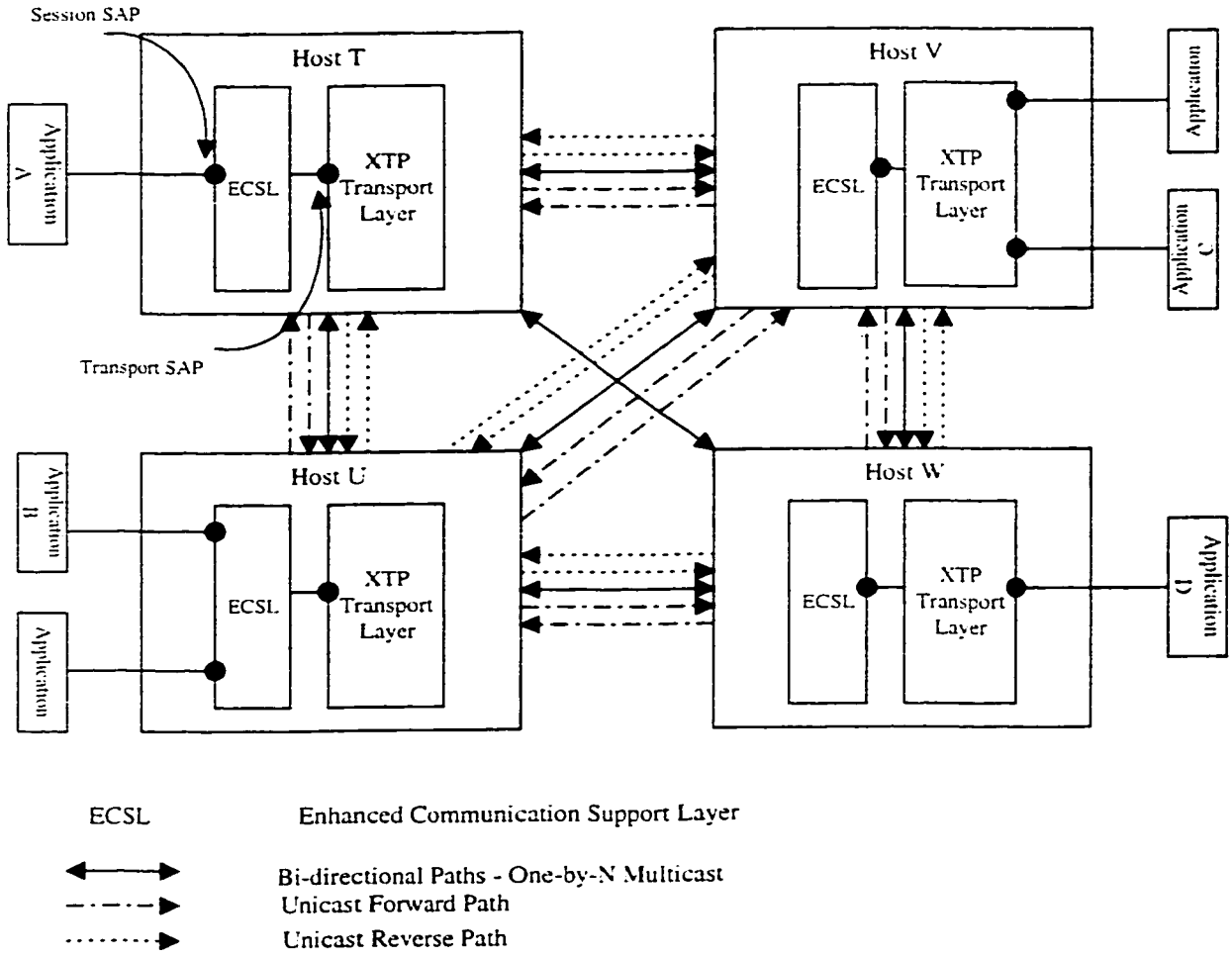

Figure 49: Ramasivan's Proposed Architecture for XTP M-to-N Communication

5.5.8 Evaluation

5.5.8.1 Strength

Our proposed architecture is a step towards building a single multicast transport protocol that can be utilized by the majority of cooperative applications. Such a protocol will be able to provide all of the major multicast features, which are reliable delivery, scalability, ordering, and M-to-N multicast. In addition, other features including supporting late joining receivers, flow and rate control and QoS can also be provided by the protocol.

A protocol that is built upon the architecture would have sufficient reliability support. Reliability between the significant members is totally guaranteed since it is based on the sender-initiated approach. Maintaining the knowledge about the entire significant set members at each sender allows the protocol to support applications where such knowledge is a must. Yet, there is no fear of either causing implosion or processing overhead at any of the senders since the significant set is often reasonably small. Error recovery within the significant set is done very efficiently since it is utilizing both ACK and suppressive NACK approaches.

Error recovery within the local groups is very efficient, since local error recovery is always attempted first. This significantly reduces both network traffic as well as processing overhead at the senders.

M-to-N multicasting is achieved with a minimal use of resources and address space. Only one multicast address, the global multicast address, is needed for data transmission. An additional multicast address, the significant set address, is used to provide an efficient error recovery scheme.

A proper ordering mechanism can easily be implemented by any protocol built upon the architecture to support any needed ordering policy by an application. This makes the protocol very suitable to all applications that require ordering, even if the ordering requirements by these applications vary broadly.

Finally, the architecture can easily be incorporated into most of the existing multicast protocols, so there is no real need to build a new multicast protocol based on the architecture. However, if desired, the architecture can easily be used to build any brand new reliable multicast protocols.

5.5.8.2 Weakness

Utilizing one multicast address per local group might be viewed as a weakness of the architecture. However, this may not be considered as a real weakness, since the advantages produced by this design choice do indeed overwhelm its disadvantages. Utilizing these local group multicast addresses significantly increases the local error recovery, and reduces network traffic. In fact, the number of local groups is simply relative to the total number of receivers in a session. In other words, the number of the utilized multicast addresses is relative to the provided scalability level. Providing a high level of scalability, with more efficient error recovery and reduced network traffic is indeed more advantageous than saving a small number of multicast addresses.

Chapter 6

Summary, Conclusion and Future Work

6.1 Summary

Here we summarize the research presented in this thesis. In Chapter 1, we introduced the major techniques of network communication, unicasting and multicasting. Although unicasting is the most competent approach for point-to-point communication, it is extremely inefficient for group communication. Multicasting however, provides very powerful mechanisms for group communication, which significantly reduce network traffic and bandwidth consumption.

In chapter 2, we introduced the major entities of a multicast session. The chapter explored some of the major multicasting issues, such as the different modes of multicasting, hardware multicasting, group membership management, and IP multicast routing. Some of the major multicast routing protocols have then been introduced.

Chapter 3 explored in depth the concept of reliable multicasting. In addition, the chapter examined other major multicasting issues such as scalability, ordering, flow control, late-joining receivers and failure recovery.

Due to the importance of multicasting, many transport layer protocols were developed to provide reliable multicast communications. Chapter 4 examined the major reliable multicast protocols, which are: the Tree-based Multicast Transport Protocol (TMTP), the Scalable Reliable Multicast (SRM), the Reliable Multicast Transport Protocol (RMTP), the Reliable Adaptive Multicast Protocol (RAMP), the Reliable Multicast Protocol (RMP), the Multicast Transport Protocol (MTP-2), the Local Group based Multicast Protocol (LGMP), and the Xpress Transport Protocol (XTP).

Although various reliable multicast protocols exist, none of them was able to provide all of the major multicast requirements. The main reason behind this significant limitation is due to the way each of these protocols has been structured. In chapter 5, we have presented a new architecture for reliable multicast protocols. With this architecture, a protocol should be able to efficiently provide all of the needed multicast requirements at the same time. A main flexibility that the architecture provides is the ability to be incorporated into most of the major existing reliable multicast protocols. Chapter 5 includes a full description of the architecture, as well as these associated issues.

The essential conclusion that we may draw from all the work described in this thesis is that achieving one standard reliable multicast protocol that can be used to serve a very wide diversity of applications is feasible, if the protocol is built upon a proper architecture. This architecture is the exact core of this thesis.

6.2 Future Work

The work described in this thesis is just a step towards building a unique standard reliable multicast protocol. Indeed, there is much work still to be done. Future work, however, could be broadly categorized as follows: evaluating the architecture in some formal way, enhancing the architecture, adopting the architecture into some of the existing multicast protocols, and possibly implementing a new enhanced multicast protocol based on the architecture.

Evaluating the architecture in some formal should be the very first step of any future work. It should be clear that our evaluation to the work presented in this thesis is merely theoretical. A formal evaluation must be done, possibly over a protocol prototype, to prove/disprove the benefits of the architecture. The architecture should be tested using different member sizes, applications loads, and available network resources. Once this is done and the architecture proves its benefits, further future work can be done to enhance and use the architecture.

Enhancing the architecture would mainly examine different ways to allow more efficient communication, as well as use of network resources. For example, research could be conducted to examine whether the same communication requirements could be achieved using a smaller number of multicast addresses.

Incorporating the architecture to an existing protocol and analyzing the benefits of such incorporation is indeed a very interesting subject of a future work. However, depending on the complexity of the protocol and the number of needed changes to the protocol, this work may need to be performed as a cooperative project.

Finally, building a brand new enhanced reliable multicast protocol that is based on our presented architecture represents a very interesting subject of future work. Indeed, we believe that such work could be a significant step towards building a standard reliable multicast protocol, which might eventually represent a great achievement in the field of multicasting.

Bibliography

[Armitage, 1996] G. Armitage. *Request for Comments 2022 "Support for Multicast over UNI 3.0/3.1 based ATM Networks", 1996.* Available at: http://www.cis.ohio-state.edu/htbin/rfc/rfc2022.html

[Armstrong/Freier, 1992] S. Armstrong, A. Freier, and K. Marzullo. *Request for Comments 1301 "Multicast Transport Protocol", Internet Engineering Task Force (IETF), February 1992.* Available at: http://www.cis.ohio-state.edu/htbin/rfc/rfc1301.html

[Atwood, 1996] J.W. Atwood, O. Catrina, J. Fenton, and W. Strayer. *"Reliable Multicasting in the Xpress Transport Protocol", 1996.* 21st Conference on Local Computer Networks, Minneapolis, Minnesota, October 1996. Conference Proceedings, pp. 202—211. Also available at: http://www.cs.concordia.ca/~faculty/bill/papers/lcn96.html

[Bormann/Ott, 1994] C. Bormann, J. Ott, H. C. Gehrcke, T. Kerschat and N. Seifert. *"MTP-2: Towards Achieving the S.E.R.O. Properties for Multicast Transport", 1994.* Proceedings of International Conference on Computer Communications Networks, San Francisco, California, September 1994.

[Cain/Deering, 1999] B. Cain, S. Deering, A. Thyaparajan. *"Internet Group Management Protocol IGMP, version 3", 1999.* INTERNET-DRAFT (draft-ietf-idmr-igmp-v3-01.txt). Available at: http://www.ctron.com/support/internet/Internet-Drafts/draft-ietf-idmr-igmp-v3-01.txt

[Callahan/Montgomery, 1996] J. Callahan, T. Montgomery, and B. Whetten. *"High-Performance, Reliable Multicasting: Foundations for Future Internet Groupware Applications", 1995.*

[Casner, 1993] S. Casner. *Frequently Asked Questions on the Multicast Backbone (MBONE)", 1993.* Available through: ftp.isi.edu:mbone/faq.txt

[Clark, 1990] D. Clark, and D. Tennenhouse. *"Architectural Considerations for a New Generation of Protocols", 1990.* Proceedings of ACM SIGCOMM, page 201 – 208, September 1990.

[Fenner, 1997] W. Fenner. *Request for Comments 2236 "Internet Group Management Protocol, Version 2", 1997.* Available at: http://www.cis.ohio-state.edu/htbin/rfc/rfc2236.html

[Floyd/Jacobson, 1996] S. Floyd, V. Jacobson, C. Liu, S. MaCanne, and L. Zhang. *"A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing", 1996.* IEEE/ACM Transaction on Networking, November 1996. Also available at: http://www.aciri.org/floyd/srm.html

[Garcia-Molina, 1991] H. Garcia-Molina, and A. Spauster. *"Ordered and Reliable Multicast Communication", 1991.* ACM Transactions on Computer Systems, vol. 9, page 242-271, August 1991.

[Hedrick, 1988] C. Hedrick. *Request for Comments1058 "Routing Information protocol", 1988.* Available at: http://info.internet.isi.edu:80/in-notes/rfc/files/rfc1058.txt

[Hofmann, 1996] M. Hofmann. *"Adding Scalability to Transport Level Multicast", 1996.* Proceedings of Third COST 237 Workshop – Multimedia Telecommunications and Applications, Barcelona, Spain, page 25-27, November 1996. Also available at: http://www.telematik.informatik.uni-arlsruhe.de/~hofmann/lgc/

[Hofmann, 1997] M. Hofmann. *"Enabling Group Communication in Global Networks", 1997.* Proceedings of Global Networking '97, Calgary, Alberta, Canada, June 1997. Also available at: http://www.telematik.informatik.uni-karlsruhe.de/~hofmann/lgc/

[Holbrook/Singhal, 1995] H. Holbrook, S. Singhal, and D. Cheriton. *"Log-Based Receiver Reliable Multicast for Distributed Interactive Simulation", 1995.* SIGCOM '95 Cambridge, MA USA, 1995.

[Johnson/Johnson, 1996] V. Johnson, and M. Johnson. *"Introduction to IP Multicast Routing", 1996.* Available at: http://www.ipmulticast.com/community/whitepapers/introrouting.html.

[Kasera/Kurose, 1998] S. Kasera, J. Korose, and D. Towsley. *"A Comparison of Server-Based and Receiver-Based Local Recovery Approaches for Scalable Reliable Multicast", 1998.* IEEE INFOCOMM, March 1998. Also available at: http://www.tascnets.com/publications.htm

[Kasshoek, 1992] M. Kasshoek. *"Efficient Reliable Group Communication for Distributed Systems", 1992.* Available at: http://www.cs.vu.nl/vakgroepen/cs/amoeba_papers.html

[Koifman/Zabele, 1996] A. Koifman, and S. Zabele. *"RAMP: A Reliable Adaptive Multicast Protocol", 1996.* IEEE INFOCOM '96, San Francisco, CA., March 1996. Also available at: http://www.tascnets.com/mist/doc/RAMP.html

[Laubach, 1994] M. Laubach. *Request for Comments 1577 "Classical IP and ARP ovet ATM", 1994.* Available at: http://www.iihe.ac.be/rfc/rfc1577.txt

[Lin/Paul, 1996] J. Lin, and S. Paul. *"RMTP: A Reliable Multicast Transport Protocol", 1996.* Proceedings of IEEE INFOCOM, 1996.

[Macker/Klinker, 1996] J. Macker, J. Klinker, and M. Corson. *"Reliable Multicast Data Delivery for Military Networking", 1996.* IEEE MILCOM '96 Conference, March 1996.

[Montgomery, 1994] T. Montgomery." *Design, Implementation, and Verification of the Reliable Multicast Protocol", 1994.* Master's Thesis, West Virginia University, 1994. Also available at: http://research.ivv.nasa.gov/RMP/

[Oosthoek, 1997] S. Oosthoek. *"Survey of Multicast Support for IP over ATM for Implementation Purposes", July 1997.* Available at: http://www.huygens.org/people/oosthoek/thesis1

[Ott/Bormann, 1994] J. Ott, and C. Bormann. *"Integrating Point-to-Point and Multicast Transport", 1994.* ICCS 94. Also available at: http://www.user.cs.tu-berlin.de/~nilss/som/som.htm

[Paul/Sabnani, 1996] S. Paul, K. Sabnani, J. Lin and S. Bhattacharyya. *"Reliable Multicast Transport Protocol RMTP", 1996.* IEEE INFOCOM '96. Also available at: http://hill.lut.ac.uk/DS-archive/MTP.html

[Petitt, 1996] David G. Petitt. *"Solutions for Reliable Multicasting"*, September 1996. Available at: http://web.nps.navy.mil/~seanet/

[Rajagopalan, 1992] B. Rajagopalan. *"Reliability and Scaling Issues in Multicast Communications", 1992.* Proceedings SIGCOM '92 Conference, ACM, page 188-198, 1992.

[Ramasivan, 2000] G. Ramasivan. *"Enhanced Communication Services for Many-to-Many Multicasting using XTP", 2000.* Master's Thesis, Department of Computer Science, Concordia University, Montreal, Canada, January 2000.

[Recommendation X.6] Recommendation X.6 *"Multicast Service Definition", Helsinki, March, 1993.*

[Talpade/Ammar, 1997] R. Talpade, and M. Ammar. *Request for Comments 2149 "Multicast Server Architecture for MARS-based ATM Multicasting", 1997.* Available at: http://www.it.kth.se/docs/rfc/rfcs/rfc2149.txt

[Towsley/Kurose, 1997] D. Towsely, and J. Kurose. *"A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols", 1997.* IEEE JSAC, Vol. 15, No. 3, April 1997. Also available at: http://www.tascnets.com/mist/doc/pub/papers.html

[Waitzman/Partridge, 1988] D. Witzman, C. Partridge, and S. Deering. *Request for Comments 1075 "Distance Vector Multicast Routing Protocol", 1988.* Available at: http://www.it.kth.se/docs/rfc/rfcs/rfc1075.txt

[Wei/Estrin, 1999] L.Wei, D. Estrin, and D. Farinacci. *"Protocol Independent Multicast-Sparse Mode (PIM-SM)", October 1999.* Available at: http://www.ietf.org/internet-drafts/draft-ietf-pim-v2-sm-01.txt

[Wright/Stevens 1995] G.Wright, and W. Stevens. *"TCP/IP Illustrated, Volume 2", 1995.* Addison-Wesley publication, ISBN 0-201-63354-X.

[XTP Forum, 1995] XTP Forum." *Xpress Transport Protocol Specification, XTP Revision 4.0",1995.* XTP Forum Inc., Santa Barbara, California, 1995. Available at: http://www.ca.sandia.gov/xtp/

[Yavatkar/Griffioen, 1995] R. Yavatkar, J. Griffioen, and M. Sudan. *"A Reliable Dissemination Protocol for Interactive Collaborative Applications", 1995.* Available at: http://www.dcs.uky.edu/~griff/papers/tmtp-mm95/main.html