ENHANCED COMMUNICATION SERVICES FOR MANY-TO-MANY MULTICASTING USING XTP

GANESH RAMASIVAN

A THESIS
IN
THE DEPARTMENT
OF
COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE
CONCORDIA UNIVERSITY
MONTRÉAL, QUÉBEC, CANADA

MARCH 2000

0-612-47850-5

Abstract

Enhanced Communication Services for Many-to-Many Multicasting using XTP

Ganesh Ramasivan

There is a strong application demand for reliable multicast. Widespread use of the Internet makes the economy of multicast transport attractive. The current Internet multicast model offers best-effort many-to-many delivery service but no guarantees. One-to-many and many-to-many services will become more important in the future. Reliable multicast transports add delivery guarantees, not necessarily like those of reliable unicast TCP, to the group-delivery model of multicast. To meet this growing demand for reliable multicast, there is a large number of protocol proposals. Because of the complexity of the technical issues, and the abundance of proposed solutions, we wish to put into place a classification of the various aspects involving reliable multicasting.

XTP, one of the well-known transport protocols, provides a toolkit of mechanisms over which the user may build the protocol he so desires. It has been observed that the multipoint-to-multipoint mechanisms as described in the appendix to the XTP specification are not very efficient. Hence, we also wish to provide a few suggestions on how to improve the existing mechanisms.

Acknowledgments

I thank the Lord Almighty for giving me the strength and perseverance to complete this work. I am forever indebted to my parents who have showered me with their love, support and encouragement since childhood. I will never forget the day my sister persuaded me to take up Science instead of Accounting. Thanks Sis!!

Thank you Dr. Atwood for giving me a chance to delve into such an interesting area of research and pointing me in the right directions whenever I felt lost. Thank you Sir, for giving up parts of your weekends for our meetings.

I would also like to thank Centre de Recherche Informatique de Montreal (CRIM) and especially my co-supervisor Dr. Petre Dini for providing me with financial assistance when I needed it most. My humble gratitude goes to my friend and counsel Dr. Francois Begin who provided invaluable suggestions to my work.

I am indebted to my present employer, Computing Devices Canada, and my managers for giving me the flexibility to manage both my thesis and my work.

A special thanks to my cousins Muthu Nagarajan and Sangeeta Iyer, the only family I have in Canada. Finally I would like to thank all my friends who have made life very enjoyable in Montreal. A special word of thanks to all the people who put me up at their place every weekend during the time I was writing up my thesis.

I would be failing in my duty if I don't express my gratitude to all the people behind the scenes, the staff and the system analyst pool in the Department of Computer Science for their assistance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the advent of the ubiquitous Internet, there is a great dependence on and a demand for computer networks to provide vast amounts of bandwidth. In its early years the "Net" was essentially used for e-mail and Usenet news. The present day Internet carries heterogeneous traffic ranging from voice and video to various types of data. There exists a multitude of requirements posed by the set of applications that generate such traffic.

Although computer networks were originally designed for point-to-point applications, there is an increasing demand for using today's networks to serve a new breed of bandwidth hungry multimedia and multipoint applications. These group applications involve multiple message exchanges among more than two application processes. This type of communication is termed as Multi-party Data Flows (MDF) and the application processes that generate these MDF are known as process groups. The advantage behind using process groups is the encapsulation of internal state and concealment of group member interactions in order to provide a distributed service with a uniform external interface.

A group communication service is a set of fault-tolerant, network communication protocols that enable the application processes to maintain a consistent

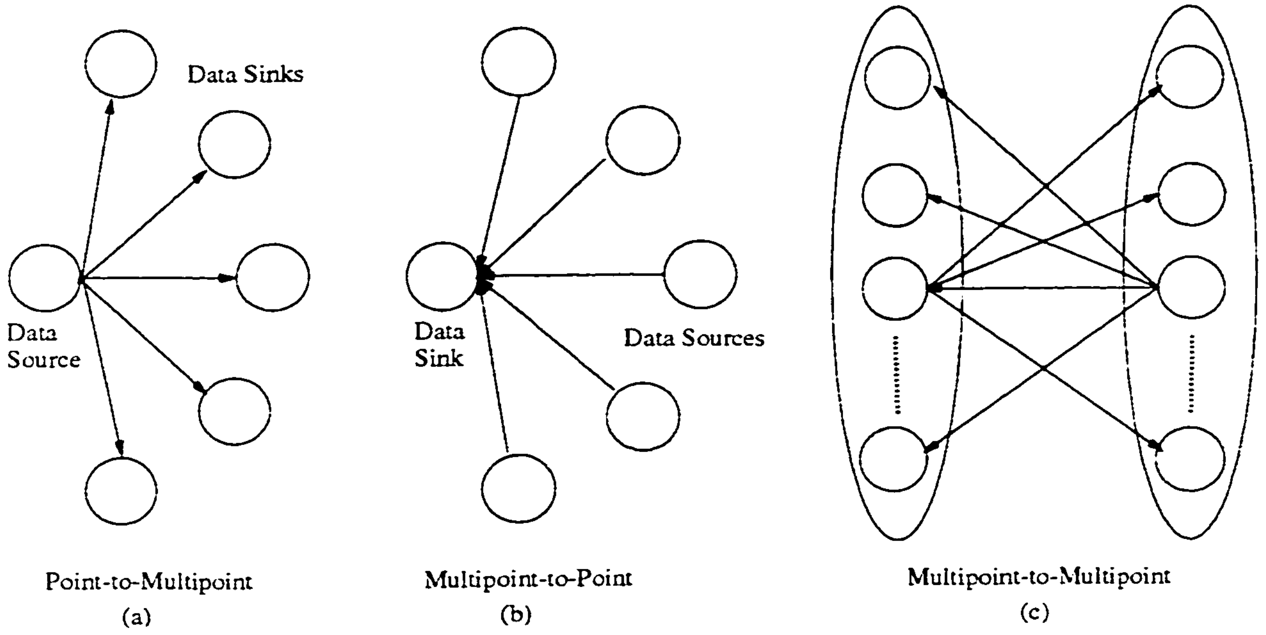


Figure 1: Multiparty Dataflow Topologies

replicated state despite random communication delays and communication or processor failures. Several existing and emerging applications, both in the local-area and wide-area environments, make use of this service including distributed-database applications, client-multiserver arrangements, multimedia applications, teleconferencing and video conferencing, and industrial automation and process control to name a few.

Multiparty data flows have higher and more complex concerns for reliability and thus they include semantics for partial failure modes resulting from dynamic group membership and its relation to message delivery. A reliable group communication service must know how and when the binding of a message to its destination receiver set occurs, must have a snapshot of its group membership at any particular point in time, detect unsuccessful message delivery and coordinate changes in group membership with message delivery. Ordered message delivery is another issue in most distributed applications. The service must provide atomic message delivery to a group with ordering guarantees.

Topologies (Figure 1) for MDF include:

1. a single data source to a set of data sinks (point-to-multipoint)
2. multiple data sources to a single data sink (multipoint-to-point)
3. multiple data sources to a set of data sinks (multipoint-to-multipoint)

The communication subsystems on these topologies are most suited to one of the three techniques for transfer of data namely unicast (one-to-one), broadcast (one-to-all) and multicast (one-to-many). When identical data needs to be transfered to a multiparty group, there are considerable efficiency and performance gains to be had by exploiting the physical broadcast media, relative to a comparable series of point-to-point transfers. Thus MDF and the complexities involved in group communication are best addressed using multicast primitives. The next chapter takes a detailed look at what multicasting really is, and it also explores the different functionalities at the various layers of the protocol stack in order to achieve true multicasting. There is also a section on how multicasting is achieved on the present day Internet.

1.1 Aims and Motivation

The current Internet multicast model offers a best-effort many-to-many service but does not offer any delivery guarantees. There is a strong application demand for reliable one-to-many and few-to-few services. Reliable multicast transport provides the necessary guarantees to the group-delivery model, but they are quite different from those available for a reliable unicast transport such as TCP.

To meet the growing demand for reliability in the multicast domain, researchers have proposed a large number of protocols. Like many research areas, there is great interest in multicasting and consequently, there arise numerous issues and problems that need to be tackled. In particular, the multipoint nature of this domain creates a whole new arena of highly complex issues. Because of such complexity, and the abundance of proposed solutions, it would be worthwhile to create a taxonomy of the reliable multicast paradigm. Several aspects of reliable multicasting make such a taxonomy particularly challenging. First, the requirements of the application at hand drive the requirements for reliability, and hence the design of the protocol. Thus, the meaning of reliability varies (as we shall soon see) in the context of different applications. Secondly, if special care is not taken in the design of reliable multicast transport protocols, the distributed nature of the multicast model can cause potential havoc to the operation of today's global Internet.

In the past, many reliable multicast protocols have been designed and implemented and there exist numerous publications that demonstrate the protocols' respective performance and suitability. However, little work has been done to compare and contrast the various features and mechanisms provided by these protocols. Hence, the first goal of this thesis, which takes the shape of Chapter 3, is an exhaustive look at these and various other aspects of reliable multicast protocols and applications and their classification into a taxonomy. An attempt has also been made to complement existing work by adding to the building blocks concept, which deals with the commonality of various components in existing multicast protocols so that they could be re-used across multiple protocols. Finally a validation of the taxonomy has been done by reviewing four well known multicast transport protocols namely the Reliable Multicast Transport Protocol (RMTP), the Local Group based Multicast Protocol (LGMP), the Xpress Transport Protocol (XTP) and the Multicast Transport Protocol (MTP), and mapping their functionalities over the taxonomy.

RMTP and LGMP mainly cater to point-to-multipoint data-distribution type of applications. They require receivers to take responsibility for data reliability and use heirarchical mechanisms to achieve scalability. But these protocols do not offer group reliability and management, leaving that task up to the application. The MTP family of protocols (namely MTP, MTP-II and MTP/SO) on the other hand, provide support for multipoint-to-multpoint group collaboration type of applications. Apart from requiring receivers to handle data reliability, these protocols also acquire and maintain information about the members of the group. MTP/SO also makes use of heirarchy to achieve better scalability.

XTP joins this list of reliable transport protocols by being able to support both the data distribution and group collaboration type applications. The fact that distinguishes XTP from the other protocols is its unique approach to protocol design. Traditionally, protocol design has involved specification of a service interface and construction of a protocol to satisfy those requirements. On the other hand, XTP provides a set of mechanisms over which a user may build the set of policies. This feature allows XTP to provide support for both unicast and multicast applications. XTP Version 3.6 [XTP3.6] specified a protocol which required receivers to take care of data reliability and did nothing for group reliability. The multicast mechanisms were then redesigned and respecified in XTP Version 4.0 [XTP4.0]. Further revisions were made to the multicast mechanisms and they became part of an Addendum to the XTP Version 4.0 specification, which was later incorporated into XTP Version 4.0b [XTP4.0b]. Despite having well defined mechanisms for point-to-multipoint communication, the current version as specified has some deficiencies with the multipoint-to-multipoint communication mechanisms. Hence, a major contribution of this thesis, the essence of which has been described in Chapter 4, is an through understanding of the problems preventing XTP from achieving efficient multipoint-to-multipoint communication, and proposal of certain modifications to the recommendations found in the addendum of the XTP protocol specification.

Finally, the last chapter presents our conclusions, summarises the contributions of this thesis, and outlines future research directions.

Chapter 2

The Multicast Paradigm

Unicasting copies of the data to a set of receivers proves to be an inefficient mechanism in cases where data needs to be transmitted from a single source to multiple destinations. A better alternative would be to multicast data whereby the source just sends a single copy of the data to all the destinations.

When processes on two or more networked computers exchange messages, there are several layers of communicating protocols at each host that work together to achieve successful message delivery. The Internet protocol architecture is divided into several layers, namely the application layer, transport layer, network layer, data link layer and the physical layer. This chapter details the capabilities of the multicast service and looks at how efficient and high-performance group communication is achieved at the physical and data-link layers and the network layer. There is also a discussion on Internet and ATM-based multicasting.

2.1 Multicast Service Model

The multicast service allows a set of hosts to participate in real-time data transfer via a multicast call. A multicast call is defined to be the relationship between these participants during data transfer. It may be simplex, where one or more senders are

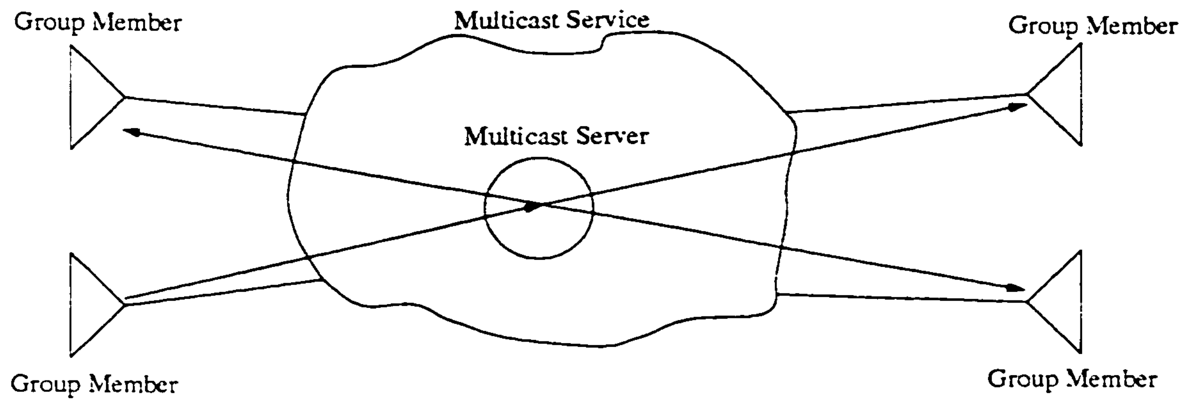


Figure 2: Multicast Service Model

defined and the rest of the participants are strictly receivers, or duplex where there may be return traffic from the receiver end in the same multicast call.

The participants of the multicast communication make use of use an intermediate logical entity called a multicast server (Refer figure 2), which provides the multicast service. The multicast server may be single or distributed, and may reside inside or outside a network, or may be on the same network as the members accessing it or on different networks.

The service also provide signals and control functions, such as multicast group creation, group membership control, call establishment and termination and provision of status information.

2.1.1 Service Capabilities

A very general multicast service was defined by the International Telephone and Telegraph Consultative Committee (CCITT) in [X6Rec93]. The following subsections provide a summary of the capabilities proposed.

2.1.1.1 Multicast Group Creation and Membership Control

A multicast group is defined as a group of members who use the multicast service to participate in one or several multicast calls. It is created and controlled by a member, who is then designated as the multicast group controller. Once created, a multicast group is assigned a unique identifier, called a Group Identifier(ID). The Group ID must be unique within a network, or it may be combined with the network address. The membership of the group is controlled by the group controller who is authorised to add (or remove) members from a multicast group.

2.1.1.2 Multicast Call

Data transfer between the members of a multicast group is through a multicast call. There may exist rules concerning exclusion (or inclusion) of members of a multicast group from a multicast call. The service should specify whether it will invite members to join a call or whether the members themselves send requests to the service to join a call. Priorities may be assigned for making calls and for transferring data between members.

2.1.1.2.1 Multicast Call Identification

A multicast call is identified by a Call Identifier (ID), which is unique within a multicast group. It is either assigned by the multicast service a priori and made known to the members prior to joining the call or it may be assigned during call establishment. At times, a member may use the Call ID to rejoin a multicast call after disconnecting from it.

2.1.1.2.2 Multicast Member Capabilities

The members of a group have certain capabilities that allow them to perform certain functions for the purpose of data transfer. These capabilities may be realized only when a group member participates in a call. A member of a group may be capable of initiating multicast calls. A member with the send capability may function as a sender (in a one-way call) or sender-receiver (in a two-way call) and one with the receive capability may act as a receiver (in a one-way call) and a receiver/sender (in a two-way call). It may receive information about the participants in a particular call. A member may be capable of allowing or denying a potential participant from joining a call or it may invite a potential participant to join a call. Other capabilities include member exclusion from a call, call termination, and reception of miscellaneous control messages.

2.1.1.2.3 Multicast Call Establishment

A multicast call may be established by static administrative means, which may be similar to those used for creating point-to-point virtual circuits, or by using dynamic on-line procedures. In the later case, a create multicast call request is issued by any member authorised to initiate the multicast call. The service may generate multiple invitation to join requests to the members of a group and the members may accept or refuse the invitation. If the service does not issue join requests, then a member may join a multicast call by sending a join request to the service which in turn responds with a join confirmed or join denied. The service responds to the call initiator once a specified number of members (also called a quorum) have joined or once a timeout occurs.

2.1.1.3 Data Transfer

A multicast call enters the data transfer phase when either the call created is sent by the service or when the sender other than the call initiator, joins the call. When a call is established through static administrative means, the data transfer phase applies when the sender interface is up.

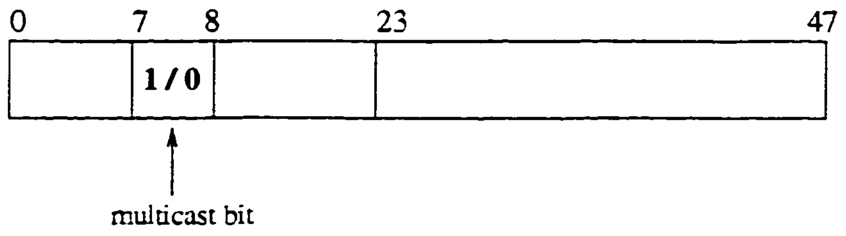


Figure 3: Ethernet Address indicating Multicast Bit

2.1.1.4 Multicast Call Termination

A multicast call may be terminated by either a member with the Terminate capability or by the multicast service due to service-specific reasons.

2.2 Multicasting at the Lower Layers

2.2.1 Hardware-based Multicasting

Multicast is well supported by shared media networks such as Ethernet, which provide efficient broadcast delivery and a large space of multicast addresses. Most networks that conform to the IEEE 802 standards provide multicast addresses that can be recognised and filtered by host interface hardware.

In the case of Ethernet, the first bit that goes on the wire determines whether the packet is a unicast or a multicast packet. The multicast bit is the least-significant bit of the most-significant byte in the vendor-id portion of the 48-bit Ethernet address. (Refer figure 3). When a sender wants to transmit multicast, it sets the multicast bit, puts an Ethernet group address in its station address portion and sends it on the wire. If any station on the subnet is not interested in this multicast, their hardware simply ignores the packet. But if a station decides to listen to this multicast, it sets a filter in its Ethernet card whereby the hardware can start accepting multicast packets belonging to this particular transmission.

The situation is more complicated in a case where the sender is attached to one sub-network while the receivers reside on different sub-networks. In this case, the handling of multicast traffic depends upon how these sub-networks are interconnected.

Devices such as link-layer bridges allow LAN functionality to be transparently extended across multiple interconnected networks, and packets can be properly forwarded to their appropriate destinations. Link-layer Switching is another technique in which incoming frames from one interface of a switch are delivered at "wire speed" out through another interface, depending on the destination MAC address in the frame header. Unlike primitive bridges and switches, which simply flood the incoming packets on all out-going interfaces, most modern bridging and switching equipment make use of efficient learning algorithms, which provide them with information about all incident branches leading to members of a given multicast group. Multicast packets are forwarded only to those segments that have interested members on them. Switches also make use of techniques such as IGMP snooping, virtual LANs and GMRP to circumvent the problems associated with flooding multicast traffic. A detailed discussion of these techniques can be found in [Gane97].

2.3 Multicasting at the Network Layer

Despite the advances in switching technologies, both at the link layer and network layer, a vast majority of the existing packet-switched networks make use of routing as a means of forwarding data across different sub-networks. This section takes a look at how multicast packets are routed across internetworks and how multicasting is achieved over the present-day Internet.

2.3.1 Multicast Groups

The set of participants in a multicast session forms a group known as multicast group. When a host wants to transmit data to the group, it need not know the other members of the group and need not itself be a member of the group. Such a group is referred to as an open group. In contrast, there also exist closed groups in which only the members of the group are allowed to send to the group. The membership of a multicast group is usually very dynamic, in that it changes over time as participants join and leave the group.

2.3.1.1 Types of Multicast Groups

Multicast groups may also be classified, based on the distribution of members across an internetwork.

- **Pervasive groups** have members on all or almost all links or sub-networks in the internetwork. These groups tend to be very long-lived and have well-known multicast addresses. Examples of such groups include widespread directory services, or net-news distribution groups.

- **Sparse groups** have members on only a small number of (possibly widely-separated) links. These groups may be long-lived or transient. Examples of sparse groups are real-time, computer-supported conferences, or replicated databases.

- **Local groups** have members on only a single link or a set of links within a single administrative sub-domain of the internetwork. These groups are highly transient and exist only as long as required for the execution of a single program. Examples of local groups are distributed parallel applications or games executing at a single site.

2.3.2 Multicast Addressing

A multicast address identifies a set of receivers belonging to a particular multicast group. Senders use this multicast address as the destination IP address of a packet

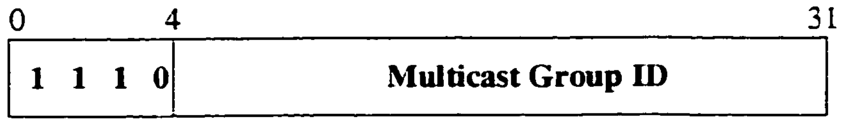


Figure 4: 32-bit Class D IP Multicast Address Format

that is to be transmitted to all the members of the group.

A multicast group on the Internet can be identified by an IPv4 Class D address that has its higher-order four bits set to "1110" followed by a 28-bit multicast group ID. (Refer figure 4). The host group address range is 224.0.0.0 to 239.255.255.255, of which some addresses are permanent and some are transient, i.e., they are dynamically assigned groups that exist only as long as they have members. The permanent addresses are assigned by the Internet Assigned Numbers Authority (IANA) which maintains a list of registered IP multicast groups (Refer table 1).

The range of multicast addresses 224.0.0.0 through 224.0.0.255 is intended for applications that never need to multicast further than one hop. Multicast routers never forward datagrams with one of these addresses as the destination, regardless of the time-to-live (TTL). The all-hosts group refers to all multicast capable hosts and routers on a physical network. Each host automatically joins this multicast group on all multicast-capable interfaces, when the interface is initialised. Membership in this group is never reported.

The IANA also owns the range of Ethernet addresses from 00:00:5e:00:00:00 through 00:00:5e:ff:ff:ff and allocates half of this block for multicast addresses. Given that the first byte of any ethernet address must be 01 to specify a multicast address, the addresses corresponding to IP multicasting are in the range 01:00:5e:00:00:00 through 01:00:5e:7f:ff:ff.

This type of allocation allows for a 32-bit Class D IP address to be mapped into a 48-bit ethernet address by placing the lower-order 23 bits of the IP address into the lower-order 23 bits of IANA's reserved block address (Refer figure 5). Thus the

Address Type	Address/Range	Assigned Group
Permanent	224.0.0.0	Reserved and cannot be assigned.
	224.0.0.1	All hosts on LAN (subnet).
	224.0.0.2	All routers on LAN (subnet).
	224.0.0.3 - 224.0.0.255	Reserved for routing protocols.
	224.0.1.11	IETF-1-Audio.
	224.0.1.12	IETF-1-Video.
	239.0.0.0 - 239.255.255.255	Reserved for site-local "administratively-scoped" applications.
Transient	224.0.1.0 - 238.255.255.255	Internet-wide multicast applications.

Table 1: Permanent and Transient Multicast Addresses

mapping places up to 32 different IP groups into the same Ethernet address because the upper five bits of the IP multicast group ID are ignored.

2.3.3 Internet Group Management Protocol

A process joins a multicast group on a given interface on a given host. The host identifies the multicast group by the group address and interface. Hosts keep a table of all the groups that at least one process belongs to and a reference count of all processes belonging to that group.

The Internet Group Management Protocol (IGMP) is used by routers and hosts that support multicasting. The mechanisms in the protocol allows a host to inform its local router, when a particular application running on it wishes to receive transmissions addressed to a specific multicast group. The router also periodically queries all the subnets to which it is connected to determine if known group members are still active or interested in the transmission. The query is sent to the all-hosts group (224.0.0.1) with a TTL of 1 so that it stays within the LAN (Refer figure 6). Each host that is interested in receiving the multicast traffic sends back one report per host group to the group address, so all group members see it.

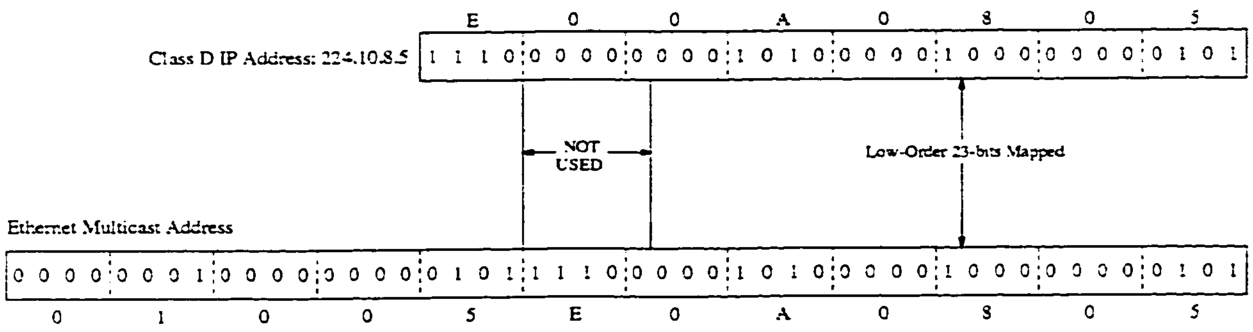


Figure 5: Mapping of a Class D IP Address into Ethernet Multicast Address

As multiple hosts may be sending reports for the same group, each host starts a random Report delay timer. If during the delay period, another Report is heard for the same group, the local host resets its timer to a new random value. Such a procedure guarantees that Reports are spread out over a period of time and that Report traffic is minimised for each group with at least one member on the sub-network.

If there are multiple routers on the LAN performing IP multicasting, one of them is elected "querier" and assumes the responsibility of querying the LAN for group members.

Based on the group membership information learned from the IGMP, a router is able to determine whether any multicast traffic needs to be forwarded to each of its "leaf" sub-networks. This information, along with multicast routing protocols, is used to support IP multicasting on the Internet.

IGMP is considered part of the IP layer and an IGMP packet is encapsulated within an IP datagram. IGMP Version 1 was specified in RFC1112 [Deer89]. IGMP Version 2, as defined in RFC2236 [Fenn97] enhances and extends the functionality provided by IGMPv1 and is also backward compatible with it. Further enhancements were made to IGMPv2 resulting in IGMP Version 3 (yet to be standardised), which is specified in an Internet draft [Cain97].

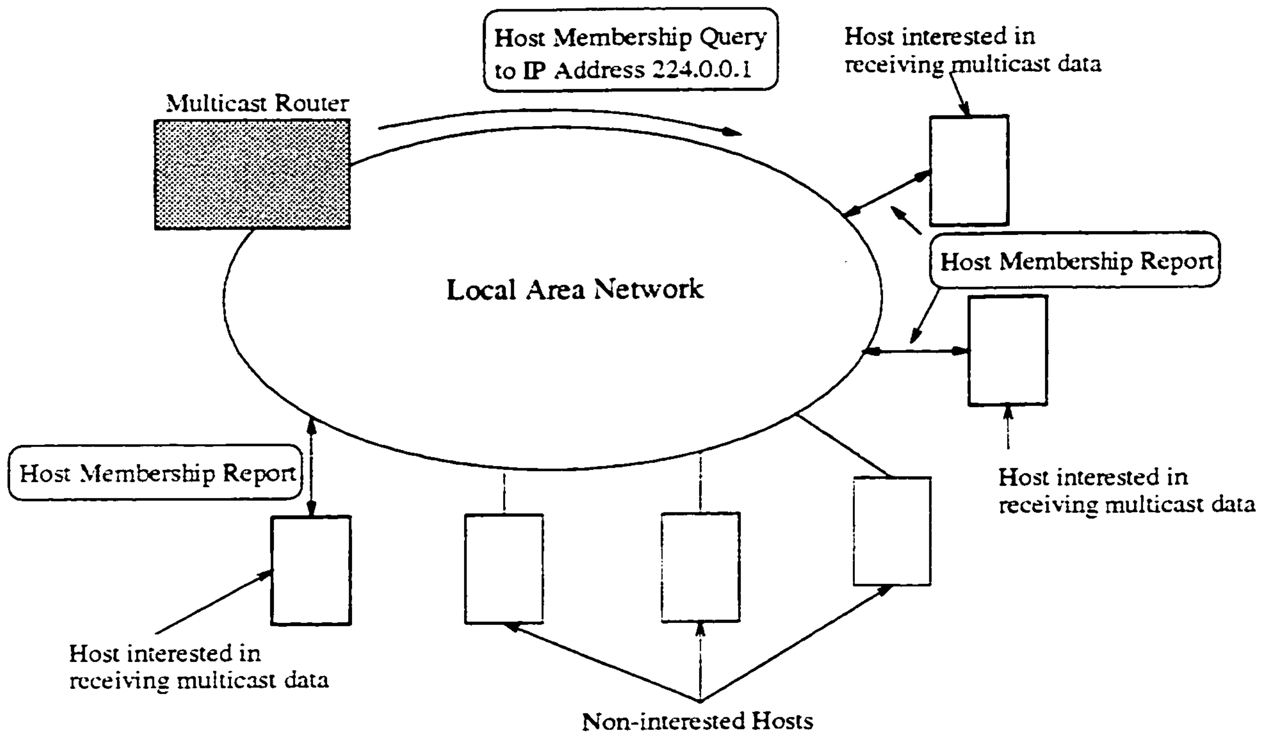

Host Membership Query to IP Address 224.0.0.1
Host interested in receiving multicast data
Multicast Router
Local Area Network
Host Membership Report
Host Membership Report
Host interested in receiving multicast data
Host interested in receiving multicast data
Non-interested Hosts

Figure 6: IGMP Messages within a LAN

2.3.4 Multicast Routing

IGMP is concerned with the forwarding of multicast traffic from the local router to group members on directly attached sub-networks. It is not concerned with the delivery of multicast packets between neighbouring routers or across an internetwork. Multicast routing protocols are necessary for such an Internet-wide delivery service. These protocols are responsible for construction of multicast packet delivery trees and for forwarding the multicast packets. Figure 8 shows how a multicast delivery tree can be constructed across multiple sub-networks using IGMP information gleaned from figure 7. Optimal delivery trees are constructed by multicast routing protocols by using a combination of potentially different multicast routing algorithms. A detailed study of the various multicast routing algorithms and protocols can be found in [Gane97].

Multicast routing through the Internet follows one of two basic approaches,

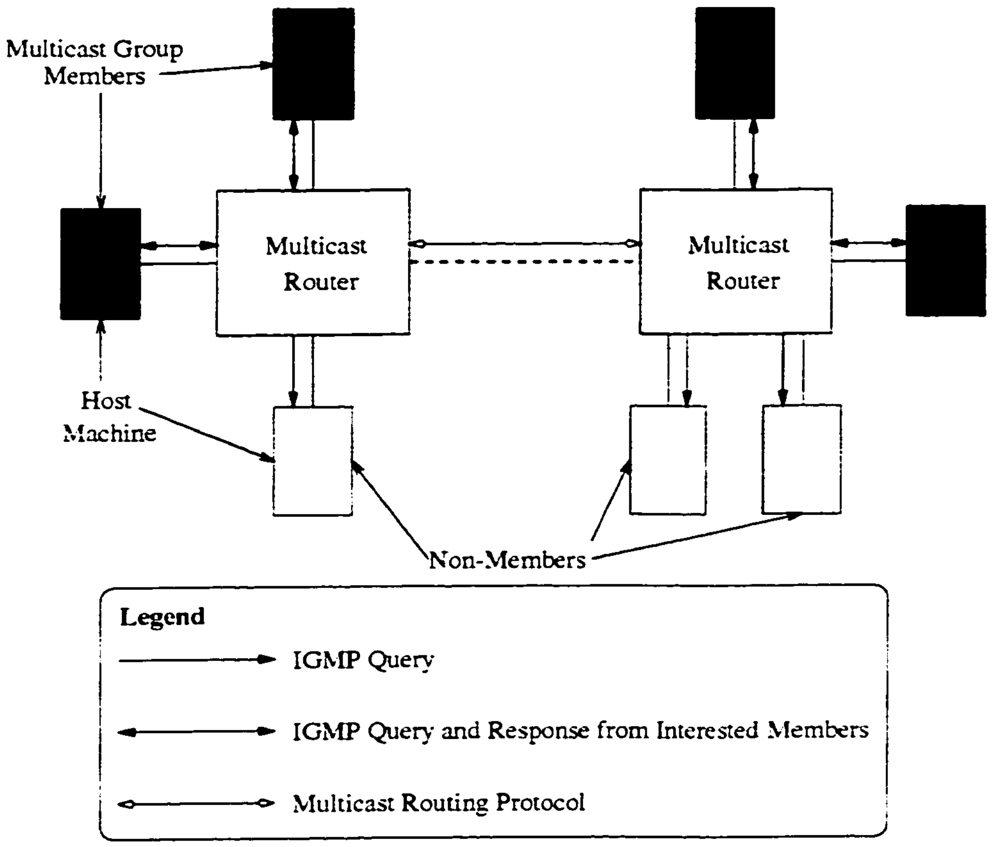


Figure 7: Multicast Routing between Sub-networks

depending on the expected distribution of receivers throughout the network. The first approach is based on the assumption that the multicast group members are densely distributed throughout the network (i.e., many of the subnets contain at least one group member) and that bandwidth is plentiful. The so-called "Dense-mode" multicast routing protocols rely on a technique called flooding to propagate information to all network routers. Dense-mode routing protocols include Distance Vector Multicast Routing Protocol (DVMRP) [Pusa98], Multicast Open Shortest Path First (MOSPF) [Moy94], and Protocol-Independent Multicast - Dense Mode (PIM-DM) [Deer97].

The second approach assumes that the receivers are sparsely distributed throughout the network and that bandwidth is not widely available. "Sparse-mode" does not imply that the group has few members, just that they are widely dispersed. In

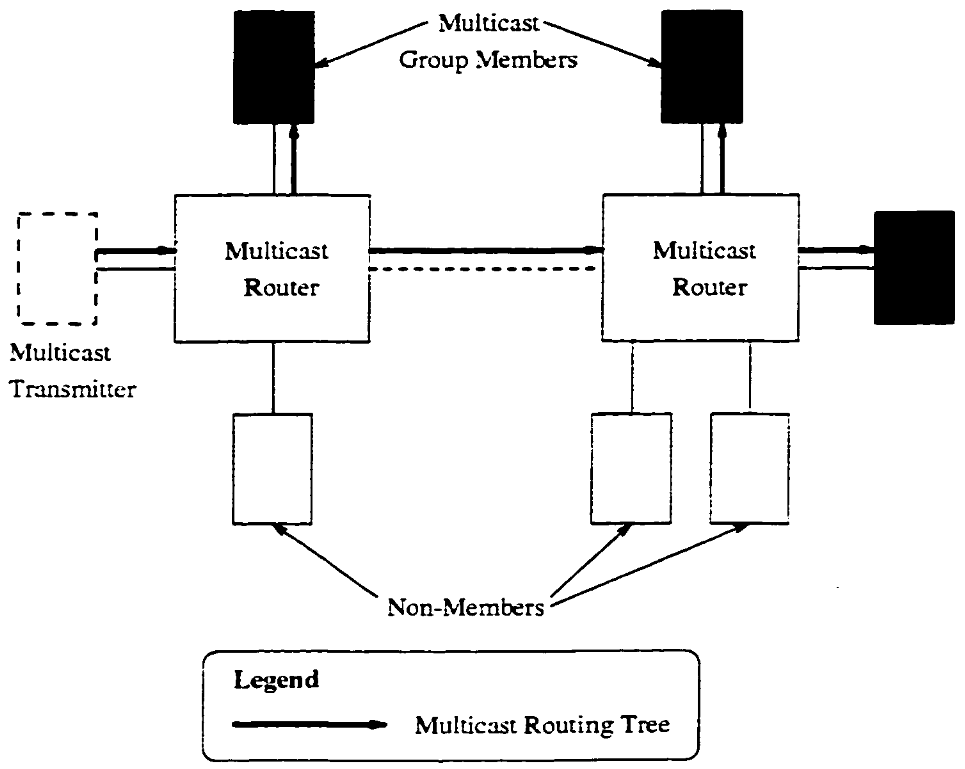


Figure 8: Construction of a Multicast Tree

this case flooding will result in wastage of network bandwidth and thus the routing protocols must rely on more selective techniques to set up and maintain multicast trees. Examples include Core-Based Trees (CBT) [Ball97] and Protocol-Independent Multicast - Sparse Mode (PIM-SM) [Deer96].

2.3.5 Internet Multicast Backbone

The Internet Multicast Backbone (MBone) is an interconnected set of sub-networks and routers that support the delivery of IP multicast traffic. It was started in March 1992, when the first audiocast on the Internet took place from the Internet Engineering Task Force (IETF) meeting in San Diego. Since then, it has been used as a testbed for deployment of several multicast applications.

The MBone [Erik94] is a virtual network, which is layered on top of sections

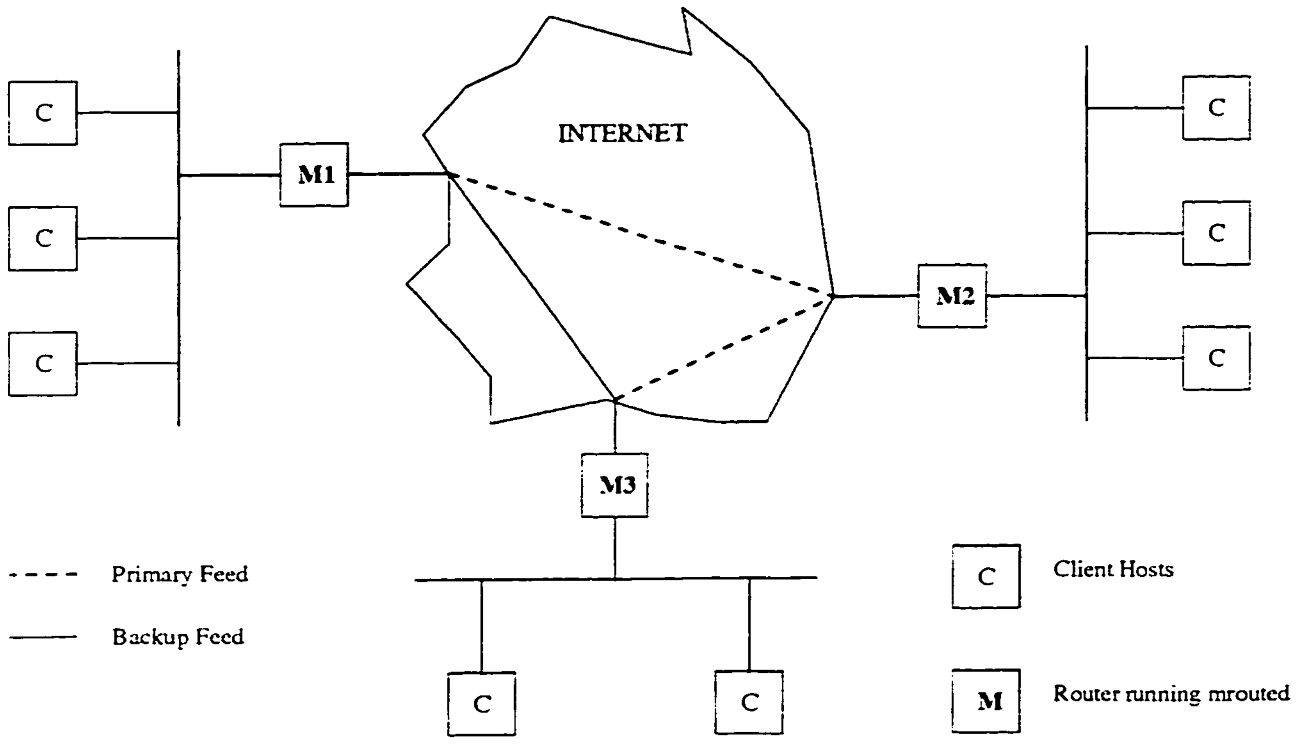


Figure 9: MBone Topology showing islands and tunnels

of the physical Internet. Since not all routers in the present Internet support multicasting, the MBone is composed of islands of multicast routing capability connected to other islands by virtual point-to-point links called 'tunnels' (Refer figure 9). These tunnels allow multicast traffic to pass through the non-multicast-capable part of the Internet. On each of these islands, there is typically a host that is running the "mrouted" multicast routing daemon, which is an implementation of the Distance Vector Multicast Routing Protocol (DVMRP).

A multicast packet will be sent from one client, who puts the packet on the local subnet, which then will be picked up by the mrouted for that subnet. The routing daemon will consult its routing tables and decide onto which tunnels the packet ought to be placed. At the other end of the tunnel is another mrouted that will receive the packet. It will examine its routing tables and decide if the packet should be forwarded onto any other tunnels. It will also check if there is any client on its subnet that has subscribed to that multicast group address and if so, put it onto the subnet to be picked up by the client application.

When a multicast packet is sent through a tunnel, it is packaged into an IP packet. There are two different methods of packaging, namely adding the Loose Source and Record Route (LSRR) IP option, and encapsulation. The older implementations of mrouted used the LSRR IP option, whereby the routing daemon modifies the multicast datagram coming from a client by appending an IP LSRR option where the multicast address was placed. The IP destination address is set to the unicast address of the mrouted on the other side of the tunnel. Due to the inherent disadvantages of this technique, most modern implementations of mrouted encapsulate the multicast datagrams. The protocol used here is IP-in-IP (IP protocol number 4), in which the original multicast datagram is put into the data part of a normal IP datagram that is addressed to the mrouted on the other side of the tunnel. The receiving mrouted will strip off the encapsulation and forward the datagram appropriately.

Each DVMRP tunnel has a metric and a threshold. These are set when the tunnel is configured. The metric specifies a routing cost that is used by DVMRP for routing the multicast datagrams. The threshold is the minimum TTL that a multicast datagram must have, to be forwarded onto a given tunnel and is used to limit the distribution scope of the multicast packets. Mrouted automatically configures itself to forward on all multicast-capable interfaces, i.e., interfaces that have the "IFF_MULTICAST flag" set (excluding the loopback interface), and it finds other mrouteds directly reachable via those interfaces.

At present, a majority of the MBone routers run DVMRP as their routing protocol, which suffers from well-known scaling problems of distance-vector routing protocols and does not (yet) support hierarchical multicast routing. As soon as more routers support "native" multicasting and tunnels can be replaced, other routing protocols such as PIM and MOSPF may be used.

2.4 IP Multicast over ATM

ATM is a connection-oriented broadcast network architecture, where end hosts must first establish a virtual channel connection if they wish to communicate. This is completely different from the IP model, where the sender relies on intermediate routers to deliver IP packets to their intended destination. However, the advantages of using ATM at the link layer has lead to the development of several multiprotocol mechanisms for encapsulating and transmitting packets using AAL5 over ATM VCs.

For multicasting over ATM, the sender should be aware of all the members of the multicast group, and only the VC root node may add or remove leaf nodes (sender-controlled). This is unlike the IP multicast model, where the sender does not need to know the receiving multicast group membership (receiver-controlled). The current ATM standards do not support the multicast group address abstraction either.

Despite these drawbacks, UNI 3.0 and UNI 3.1 ATM based networks implement multicasting using the basic point-to-multipoint service. The two most common implementations are "VC meshes" and "multicast servers". In an ATM network, the multicast-capable interfaces are grouped into "clusters", where a cluster is a set of ATM interfaces able and willing to achieve AAL level multicasting.

In the multicast VC mesh (Refer figure 10), a point-to-multipoint VC originates from each sender to all the members of the multicast group. In addition, the ATM interface must terminate one VC for each active source in the cluster. Nodes that are both sources and group members will originate a single VC and then terminate a branch of one other VC for every other node sourcing traffic to the group, thereby resulting in a criss-crossing of VC's (i.e., a VC Mesh) across the ATM network. Though the VC mesh reduces the transmission overhead on any given source node and provides transmission reliability, it requires that each source node maintain a list of all the recipients and update the list when there are changes in group

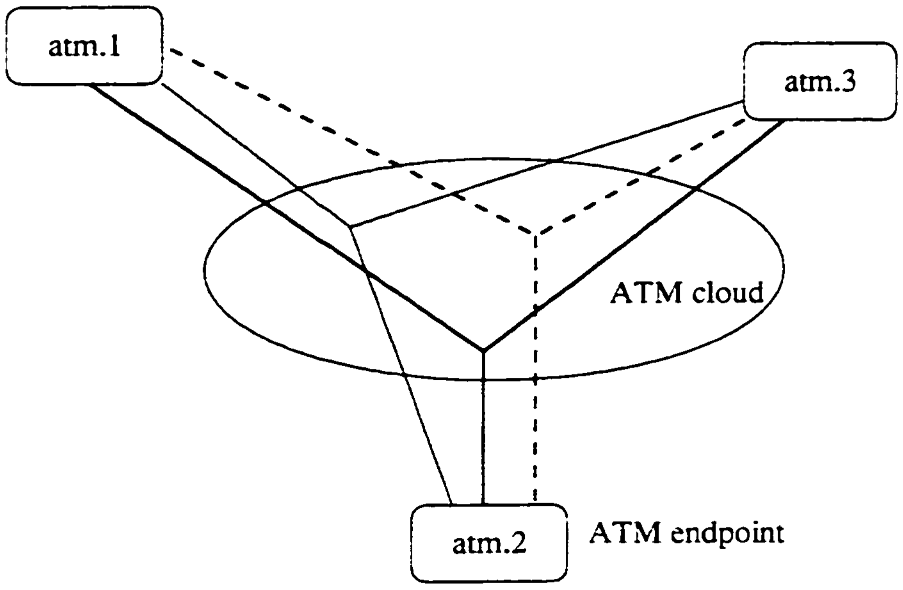


Figure 10: Multicast VC Mesh.

membership.

In the multicast server (MCS) model (Refer figure 11), a server is chosen within each cluster. The MCS establishes a point-to-multipoint VC to all the destinations and each source establishes a point-to-point VC with the MCS. This model is sometimes called a "shared-tree" model, as the traffic from all senders shares the distribution tree from the multicast server. During the data transmission phase, the MCS reassembles messages arriving on all incoming VC's and queues them for transmission on its outgoing multipoint VC. Though this method is more adapted to dynamic group membership control and resource utilisation (only two VC's are needed per ATM interface), the MCS may create a potential bottleneck since the traffic from various sources becomes concentrated on it. Although both VC meshes and MCSs provide multicast services over ATM, their inherent drawback is that the addresses of the intended recipients must be known in advance. This is not a requirement in the classic IP multicast model, since the routers know (through IGMP information), if clients on their subnet are part of a multicast group. To address this limitation, the IETF proposed to set up a Multicast Address Resolution Server (MARS) on the

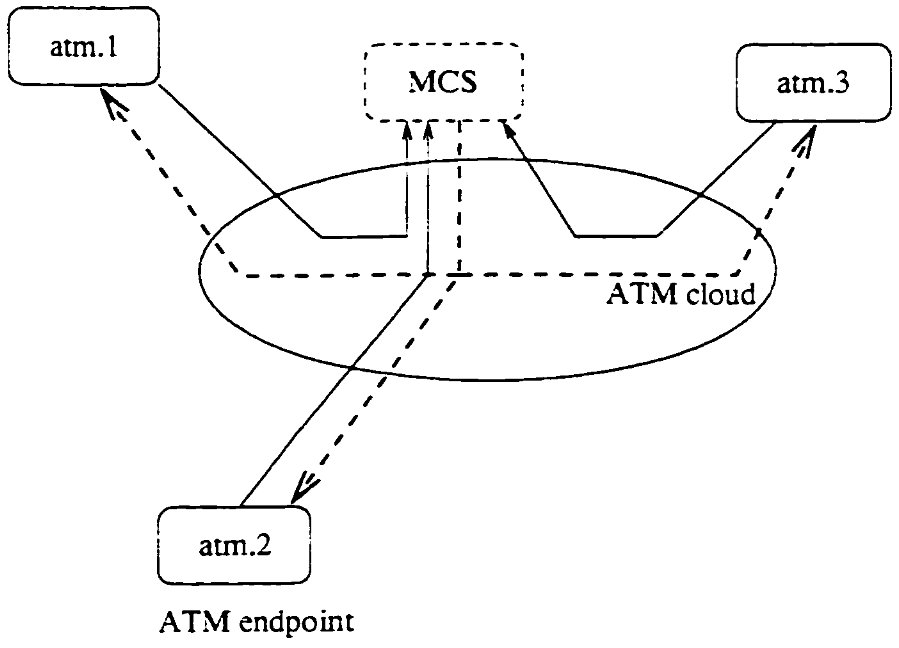


Figure 11: Multicast Server

ATM network [Armi97].

MARS is a central registry, which maintains a table of Multicast Group Address, ATM address1, ATM address 2, ..., ATM address N mappings for every Layer 3 multicast group that has one or more members. Each MARS manages a cluster of IP-ATM endpoints. Every IP/ATM interface (both host and router interfaces) which is logically attached to a particular cluster is considered to be a 'MARS client'.

Two different types of VC's are used to carry control messages between a MARS and its MARS clients. (Refer figure 12). A transient point-to-point VC to the MARS carries query/response activity initiated by a MARS client, while control messages are propagated by MARS using a semi-permanent point-to-multipoint VC that has all its clients as leaf nodes. This semi-permanent VC is also called a "ClusterControlVC". A MARS client must initially register itself as a cluster member with the MARS, allowing it to add the new members as a new leaf of the ClusterControlVC.

When a MARS client wishes to send to a multicast group, it queries MARS to see if the group exists. If it does, the MARS sends the source node a list of all

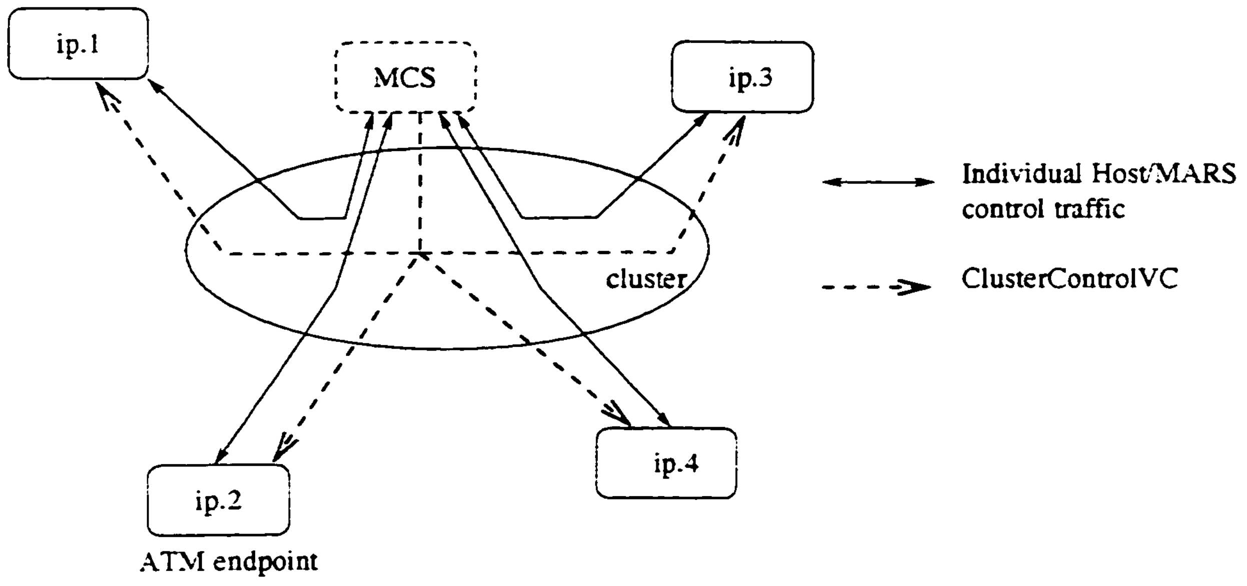


Figure 12: VC's carrying MARS Client-related Control Traffic

the addresses of destination nodes that are members of the group. If the group does not exist or MARS does not have any knowledge of it, the source is notified and the connection packet is dropped. Once the source has the list of destination addresses, it establishes a point-to-multipoint VC to the destination nodes. The MARS notifies the source node when the membership of the group to which the source is transmitting changes.

When a destination node wishes to join a multicast group, it sends a request to the MARS to include it to the group. The MARS tracks the addresses of all nodes that are members of its multicast groups, and maintains the list by periodically querying its clients.

MARS works with both VC meshes and MCS multicasting techniques, and a given MARS can support a mixture of both techniques on the same network.

Chapter 3

Reliable Multicasting

3.1 A Taxonomy of Reliable Multicast Protocols and Applications

Multicasting is defined as a transmission mechanism where a single sender transmits data to a group of receivers. The issues involved in reliably transmitting data to such a group poses a significant technical challenge. IP Multicast does not have the capability to provide the data delivery guarantees that are required by a vast majority of applications that make use of the multicast paradigm. Such delivery guarantees can only be offered by reliable multicast transport protocols that work on top of the network multicast service.

It is obvious that a multicast transport connection is required among the users of the the multicast transport service for the purpose of data transfer. The users who participate in a transport connection form a group and access multicast services through a transport service access point (TSAP)[1]. The group of transport service users maintain shared state information in order to support the mechanisms

[1]An Internet application can access multicast services using a combination of a 32-bit Class-D IP address and port number.

of the data transfer phase and is termed an "active group".

The ever growing number of reliable multicast protocols indicate the vast interest in this active research area. However, the question of what is reliable in many of these protocols gives us few precise answers. It is the skewed understanding of the term "reliable" that encouraged us to build a taxonomy of reliable multicast. Past attempts at such a taxonomy include [Diot97], which is a survey of multicast protocols in terms of functionality and mechanisms for reliable multicast transmission, and [Obra98] which reviews several existing multicast transport mechanisms and classifies them according to their distinct features. The Reliable Multicast Transport Working Group (RMTWG), which is part of the IETF, is also looking to develop a framework for standardisation of reliable multicast transport. It has published several draft standards including [Hand99], which provides a review of the design space for reliable multicast protocols intended for bulk data transfer, and [Whet99], which attempts to bring out certain components that are common to multicast protocols and standardise them as "building blocks", which may be re-used across multiple protocols.

Our taxonomy complements the existing work mentioned above and covers new territory by looking at the various issues involved in many-to-many data transfer. It also brings out the orthogonality between the various requirements of reliable multicast transport protocols and the mechanisms used to satisfy those requirements. We also add to the list of building blocks by isolating several common mechanisms amongst the profusion of multicast protocols. The motivation behind isolating these common mechanisms comes from the fact that the Xpress Transport Protocol (XTP) [Stra92] has been designed to provide a set of mechanisms over which the necessary policies may be implemented. More on XTP can be found in section 3.2.4 and in Appendix A of this thesis.

3.1.1 Classification of Requirements for Reliable Multicast Applications

3.1.1.1 Types of Reliability

The notion of reliability in multicast communication is highly complex and plays a major role in the design of the transport protocol. The term "reliable" multicast can be seen in several different perspectives and can be defined in several ways.

[Garc91] defines reliability based on three properties of a multicast protocol. The first is the length of time it takes to deliver a multicast message, either measured from message initiation at the source or from the time the first message arrives at a group member. Delivery time may be guaranteed at one group member, a majority of group members, or all group members. The second property, atomicity, requires that all group members deliver a message to the application within a specified interval begun either after one group member has delivered the same message or after a majority of group members have delivered the message. The final property is ordering, which may range from no guarantees of ordering, to a guarantee that messages are always delivered consistently even in the presence of network failures.

A less stringent definition of reliability is given in [Raja92] which defines protocols to be reliable if they offer completeness and finiteness. Completeness means that the protocol delivers multicast messages in the same order as sent by the source, without message duplication or loss. No mention is made of ordering guarantees in the presence of failures. Finiteness is similar to atomicity.

For applications with strict latency requirements, the data is only useful if it is delivered within bounded time. One issue which an application must deal with is the actions that must be taken when the data cannot be delivered reliably within the delay requirements. [Wang97] describes a way of broadening the application reliability requirement from simply reliable or unreliable to a more general spectrum, by introducing a parameter called Reliability Interval (RI). RI is defined as the time interval within which reliable delivery is useful, and serves as a way to describe the connection between reliability and latency. For instance a large RI would indicate less stringent latency requirements and a small RI would indicate the reverse. So, a mailing list distribution application may have an RI of 7 days while a multiuser conferencing tool may have an RI of only 5 minutes.

3.1.1.1.1 Data Reliability

Despite varying views on the term reliable, multicast applications can be classified based on how reliably the transport connection delivers data to the layer above it.

- **Best-Effort Reliability:** As the name suggests, best-effort reliability does not guarantee any sort of reliable delivery. The application, if required, may implement some sort of reliability mechanism. There is also the issue of loss of data packets versus the corruption of the contents of the data packets. In case of the contents of a data packet are corrupted, there will be no use for that data packet anyway and hence it is required of a best-effort protocol to transfer with the best of its ability, uncorrupted data packets. Most UDP-based IP Multicast applications fit into this category.

- **Bounded-Latency Reliability:** In this case, each packet adheres to a specified lifetime over which the data is useful to the receiver, thereby defining an upper bound on its delivery latency. Packets arriving outside this time frame are discarded. A common applications which requires such a reliability guarantee is a video stream, where each packet has a playback time and any packet not meeting this deadline is discarded.

- **Most-Recent Reliability:** This type of reliable transmission targets applications where only the most recent data of a particular parameter is of interest.

A simple example would be a service that provides reliable updates of stock quotes. If a particular stock quote is lost, and a new update is received before a retransmission can occur, the old data is rendered useless. Thus it is possible that the data may take on a value that is never known to some or all of the receivers. Other example applications include Distributed Interactive Simulations (DIS) and situational awareness dissemination in shared WAN environments.

- **Absolute Reliability:** This type of reliability requires that all the transmitted multicast packets be delivered to the active group. If any of the data is missing at the receiver even after repeated retransmissions, then none of the data will be useful. It is analogous to the reliability supported by TCP for unicast sessions. Multicast file transfer is a good example of an application requiring absolute reliability.

3.1.1.1.2 Group Reliability and Management

It should be noted that reliable multicasting not only concerns reliable delivery of data to the members of the group but also reliable knowledge and maintenance of the group membership. Point-to-point unicast transmission involves only two parties, and hence communication failure is represented by failure of either of the two parties. However, in the multicast case, the issue is less straightforward. Many protocols that exist today claim to be fully reliable, but fail to implement any sort of group reliability mechanisms. There exist a set of conditions concerning an active group that must be true in order for a transport connection to enter or remain in the data transfer phase. The term "active group integrity" or AGI, identifies and defines the necessary conditions in accordance with various levels of group reliability.

- **0-reliable Multicast:** This is the easiest way to multicast information on a network. The sender is not required to have any knowledge of its set of receivers. In this case the AGI is typically zero. Many MBone applications that exist today adopt this model.

- **K-reliable Multicast:** A multicast transmission amongst a group of N receivers is said to be K-reliable where ($0 \leq K \leq N$), if at least K members of the group are alive at a particular instant. If a receiver failure is detected, then that particular receiver can be removed from the group and the remaining group members may continue. Another semantic may require the protocol to monitor the cardinality of the group and abort the transmission if the membership of the group drops below K. A more rigorous requirement could be enforced by requiring that not only must K members be present, but they must be K specific members.

- **Atomic Multicast:** Atomic multicast requires simultaneous delivery to all members of the group or to none of them. In most cases, the transport connection needs to be terminated in case the AGI is violated. A basic atomic multicast protocol requires two phases, namely, a data transfer phase (transmission and acknowledgement), and a validation phase (notification of receipt of packet to all receivers). Once a packet is validated, receivers are allowed to deliver the packet to the application (See figure 13). The two phases do not need to necessarily run in a stop-and-wait mode as the validation may be implicitly sent in a following data packet.

A relationship between the data group reliability semantics can be derived from the above classification. For instance, one may have best-effort data delivery with none or K-reliable group semantics but it makes no sense to have atomic multicast with best-effort data delivery. Similarly absolute data reliability can be categorised with all the three types of group reliability semantics.

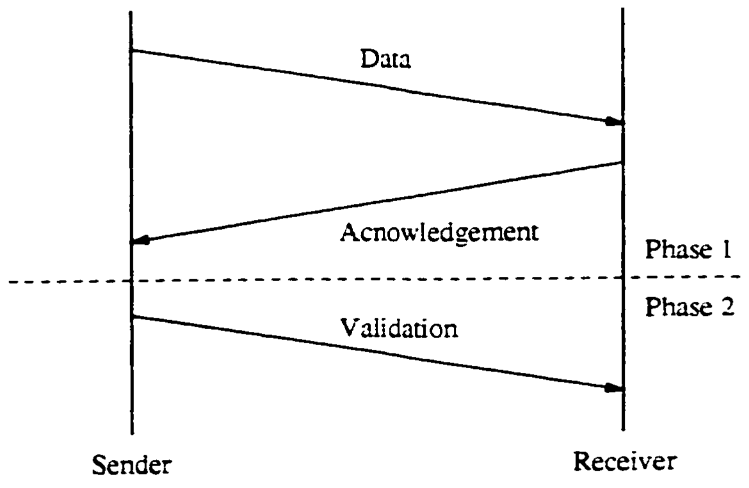


Figure 13: Two Phase Validation Protocol for Atomic Multicast

Multicast group management on the other hand, deals with managing the group in accordance with the AGI. The level of reliability provided by a multicast protocol depends on the control algorithms and group policies used for managing the group.

3.1.1.2 Multicast Data Distribution

There is a plethora of applications that require an underlying 'reliable multicast' service. A further classification of these applications can be done based on the mechanism used for data dissemination, into point-to-multipoint data distribution applications and multipoint-to-multipoint interactive applications.

Point-to-multipoint or 1-by-N multicast applications deliver data from a single source to multiple receivers, and usually run without much human intervention. A few examples for this set include software distribution, data distribution and replication, and mailing list delivery.

The latter group involves multiple sender participation (M transmitters and N receivers) and are commonly referred to as M-by-N applications. They often have direct human involvement and have stringent latency requirements due to their interactive nature. Examples include interactive distributed simulations, networked gaming and other collaborative applications such as tele-conferencing and video-conferencing.

3.1.1.3 Ordering

Packets may arrive out of sequence at their destination due to packet losses or changes in the datagram routes. Many distributed applications require an ordered reception of packets, because misordering may give a different view of the state of the group. An ordered reliable multicast communication amongst a set of TS-users is defined based on how the transport service data units (TSDUs) of a sending TS-user are presented to the receiving TS-user and how the receiving TS-user gets TSDUs from the sender.

In the case of single sender ordering, the TSDUs generated by the sending TS-user must be delivered to each receiving TS-user in the active group in the same order in which they were sent. In the case of multiple sending TS-users, ordering is determined in terms of the relative sequencing of TSDUs received from multiple sending TS-users. The ordering relationship defines the arrangement or interleaving of TSDUs from multiple senders and can be classified as no ordering, local ordering, causal ordering, partial ordering and total ordering. It should be noted that ordering properties may also apply at a protocol level and in such cases the TSDUs are replaced by transport protocol data units (TPDUs). The distinction between the two has been made specifically for this purpose. The requirements for ordering that an application may impose on a multicast protocol can be classified as follows:

- **No Ordering:** This is the simplest requirement where the TS-provider does not guarantee any relationship between TSDUs sent from a single sending TS-user or from multiple sending TS-users. Note that even though the ordering of TSDUs are not guaranteed, the ordering of TPDUs belonging to the same TSDU are guaranteed.

- **Local Ordering:** Also called FIFO ordering or single source ordering, TSDUs are guaranteed to be delivered to the receiving TS-users in the same order they were transmitted by the sender. There is no ordering rule specified for the TSDUs transmitted by different senders. It is similar to the way TCP performs ordering in case of unicast.

- **Partial Ordering:** The TSDUs, generated by all sending TS-users are delivered to each receiving TS-user according to an arbitrary ordering rule. If the TSDUs are ordered according to a rule applicable to all the receivers, then each receiver receives the TSDUs generated by all the senders in the same order. On the other hand, if the TSDUs are ordered according to a rule determined by each receiver, then the TSDUs may be received in different orders.

- **Causal Ordering:** Causal ordering requires that ordering be maintained across distributed processes. The TSDUs generated by all senders are ordered according to the causal dependence relationship among sending events. A causal dependence relationship is established between two sending events, A and B, if the following applies:

  1. A happens before B if A and B are sending events generated by the same sender and A is sent before B.
  2. A happens before B if A and B are sending events generated by two different senders and the TSDUs generated by event A by one sender is received by the other sender before it generates the event B.

  A causal dependence relationship is established among more than two sending events if it can be established that A happens before B and that B happens before C, and it therefore follows that A happens before C.

- **Total Ordering:** TSDUs transmitted by multiple senders are received in the same relative order and delivered in sequence to each receiver. This level of ordering is typically required in distributed database applications. In the case of multicasting from a single sender to multiple receivers, causal ordering and total ordering are equivalent.

A protocol that ensures, in addition to reliability, a total ordering of the delivered messages is called an atomic protocol. Such a protocol may be used for reliable validation, atomic operations, group memberships etc., while a causal protocol (ensuring causal and not total ordering) may be used for ensuring consistency in updates to replicated data.

3.1.1.4 Scalability

Several multicast applications require scalable multicast communication support for large groups having hundreds or thousands of members who undergo a very high number of group changes per second.

On the other hand, there may exist certain applications which might not require the protocol to scale to extremely large groups. For instance, if an application were to be used only over a LAN environment, the protocol could be designed more efficiently if it made use of the broadcast capability of the underlying physical media. There can also be certain practical limits imposed by the application which reduce the need for scalability. For instance, if we consider a white-board application, there is a practical limit on the number of users who can simultaneously carry on a useful session. Hence, it is highly unlikely that such an application would ever require the protocol to be scalable to even 20 users.

3.1.1.5 Random Receiver Membership

Unlike the unicast case where the absence of a receiving end-point during the establishment of a connection causes the activity to abort and an error is indicated to the higher layer, it is possible that there be no receivers during establishment of a multicast connection. Hence applications may require protocols to allow receivers to join at any point in time. Similarly, members should also be allowed to leave multicast groups at any time.

3.1.1.6 Congestion Control

Congestion control is one of the most difficult requirements to satisfy in multicast due to its multipoint nature. Uncontrolled propagation of multicast data by even just a few sources can potentially cause havoc on large-scale inter-networks. Thus, for reliable multicast to be accepted and embraced, it must address congestion control requirements in an efficient manner. These multicast protocols must also co-exist with the more common unicast protocols like TCP.

3.1.1.7 Quality of Service

QoS characteristics for a multicast application include throughput, tolerance to transit delay, transit delay jitter and loss and corruption of packets. The QoS requirements are usually specified through the selection of values of QoS parameters before a transport connection is operated. Once agreed, QoS values usually apply for the duration of a connection.

3.1.1.8 Security

As IP Multicast currently provides no security features, a multicast transmission on the Internet can be received by anyone listening on that particular multicast address and port as long as that receiver is within the TTL range of the sender. It should be recognised however that privacy and security will be required of multicast protocols, considering the multitude and types of existing and emerging multicast applications.

3.1.2 Classification of Reliable Multicast Mechanisms

Now that the requirements for most reliable multicast applications have been listed, we can proceed with classifying the mechanisms used by reliable multicast protocols to satisfy these requirements.

It is obvious that the numerous requirements in the multicast case drives the need for more complex mechanisms than what is available in unicast. Reliable multicast protocols, like their unicast counterparts, provide a variety of mechanisms for handling acknowledgements, error correction, PDU ordering and flow and congestion control. Other multicast specific mechanisms are also available for maintenance of state and group-membership information, data-distribution, delayed receiver participation, QOS, and timing. Group-coordinator mechanisms are also available for the purpose of group management.

3.1.2.1 Data Reliability Mechanisms

Multicast protocols can be grouped into two broad classes namely, sender-reliable and receiver-reliable, depending on whether the sending end or the receiving end is responsible for implementing the reliability mechanisms.

3.1.2.1.1 Sender-Reliable

This approach places the responsibility of reliable data delivery and retransmission of lost packets on the sender. The state information about all the receivers is monitored by the sender and is maintained using positive acknowledgements (ACKs) from receivers for every packet correctly received, and timers for the purpose of detecting packet losses.

The approach is appropriate in cases where the sender must have absolute control of the group (e.g., security reasons). This mechanism is most appropriate in cases where atomic multicast is required. The main disadvantage of this scheme is that it does not scale due to the overwhelming of the sender with responses from individual receivers (also known as the ACK implosion problem). The sender incurs an additional burden in maintaining the state of all receivers. Putting multiple ACKs into a single data packet reduces the implosion problem by a constant amount, but there is a limit to this technique as well.

3.1.2.1.2 Receiver-Reliable

The alternate approach shifts most of the responsibility for reliable data delivery to the receivers. Each receiver maintains reception state and requests repairs via negative acknowledgements (NACKs) when errors are detected. Error detection is based on the receiver perceiving gaps in the data. This requires that individual packets be identified using sequence numbers.

There are several classes of receiver-reliable approaches that are discussed below.

3.1.2.1.2.1 Sender-Oriented

In sender-oriented approaches, an error detection at the receiver results in a NACK to the sender. Though there may be other receivers who have received the data for which this NACK was issued, only the sender is involved in issuing repairs. This approach is appropriate when receivers cannot communicate with each other (perhaps for security reasons). However, it limits scalability due to the NACK implosion effect at the sender for a large number of receivers. Thus, it is best suited for transmission of very large data packets in low-error environments whereby a low NACK overhead to data content ratio can be realized.

3.1.2.1.2.2 Receiver-Oriented

In a receiver-oriented approach, receivers can communicate with each other to assist in error recovery. Each receiver caches data for some time or for the entire session. When a particular receiver detects an error, it multicasts a NACK to the whole group. This technique is also called "flat-receiver oriented" error recovery. In this case, any receiver that has correctly received and cached the missing data can issue the repair.

The above approach in itself would not reduce the NACK implosion effect since the NACK is sent to the whole group and any receiver which detects an error could in turn issue a NACK. To further improve scalability and allow for distributed state management and more organised repair schemes, it is possible to introduce hierarchy into reliable multicasting. Such a "hierarchical-receiver oriented" error recovery scheme may involve receivers organised into a tree structure. NACKs generated by them are aggregated by a parent node and passed on to the level further up in the hierarchy. Using appropriate router mechanisms, or with multiple multicast groups, it is also possible to allow the intermediate tree nodes to retransmit missing data to the nodes below them in the tree rather than relying on the original sender to retransmit the data.

A further improvement to the distributed repair scheme is suppression of control messages per repair. When a receiver detects an error, it is highly likely that other downstream and equidistant receivers (in terms of delay) will also experience the error. In addition, the equidistant receivers are also likely to detect the error at roughly the same time. To reduce the chance of all such receivers issuing NACKs simultaneously, each receiver sets a random timer upon error detection. When the timer expires, and if a NACK for the missing data has not been heard, then the receiver issues the NACK. When a receiver that has correctly received and cached the missing data hears a NACK, it will start another random timer. If this timer expires and a repair for that particular packet has not been heard, one is issued. This

Reliability Mode	Error Detection	Repair Responsibility	Scalability
Sender	Sender	Sender	Low: due to ACK implosion
Receiver	Receiver	See Table 3	See Table 3

Table 2: Reliable Multicast Approaches

method is helpful in reducing both the number of repair requests and repairs that might otherwise be heard.

Another method used is the deterministic suppression of responses in cases where the downstream receivers also detect the same errors as upstream receivers. By accurately estimating the delay between receivers, the uniform distribution of the downstream random timers can be adjusted to produce longer delays. Thus it is likely that a downstream receiver will observe the NACK of an upstream receiver before issuing his own NACK. Typically, a combination of randomised and deterministic NACK/repair suppression is used for a flat, receiver-oriented reliability scheme.

The advantage of tree-based approaches is high fault tolerance and scalability. Node failures affect only a subset of the receivers, each of which can easily and locally decide to bypass its parent and report directly to the node one level higher in the tree. One drawback is the amount of state information required at each node to perform the above mentioned functions. Another drawback is the requirement that all receivers must cache data for retransmission. Though this is not a problem for applications like interactive white-boards, where state is maintained regardless, it is highly disadvantageous for long-lived applications that require significant buffering.

Tables 2 and 3 summarise the reliable multicast schemes presented above.

The receiver-reliable approach imposes constraints on senders in cases where applications demand complete reliability. Since senders do not track receiver reception state, and receivers may request retransmission of data at any point in the future, the sender needs to buffer data indefinitely. This is not an issue for any applications

Repair Orientation	NACK Scheme	Repair Scheme	Scalability
Sender-Oriented	Receivers NACK to Senders	Sender issues repairs to group	Moderate: Least scalable of receiver reliable approaches
Flat, Receiver-Oriented	Receivers NACK to group	Receivers cache data and can issue repairs	High: Limits NACK implosion but requires distributed buffering
Hierarchical, Receiver-Oriented	Receivers NACK to some hierarchical node or group	Hierarchical nodes successively responsible for buffering and issuing repairs	Very high: Excellent scalability and limited network overhead

Table 3: Receiver Reliable Multicast Approaches

delivering data with a bounded latency, because the sender is free to discard data that cannot meet the reception deadline. However, it does represent a problem in cases where the validity of absolutely reliable data never expires and hence the sender cannot arbitrarily discard it.

Another problem which arises due to the use of NACK-based protocols is the fact that a sender is never sure whether a receiver is silent (not sending any NACKs) because it is correctly receiving all the data or it just has died. In such cases, the protocol must use some sort of positive acknowledgement mechanism to ensure that receivers are alive throughout the session.

There is also the issue of excessive network load caused by local retransmissions. In such cases, the protocol must have mechanisms to dynamically determine, based on the number of receivers requesting retransmission, whether the data should be multicast again to the whole group or just unicast to the individual receivers.

3.1.2.1.3 Open-Loop Delivery Mechanisms

There exists a class of applications that do not provide any sort of feedback to the transmitter. The primary example of this is satellite-based transmissions where the back channel may be very narrow or even non-existent. In such cases the solution is to take the initial data and encode it using an forward error correction (FEC)-style mechanism. This encoded data is then transmitted as a continuous stream. Receivers can then join a session and receive packets until they have received a sufficient number to decode the original data, at which point they leave the session. The advantage of this solution is that it can scale to an infinitely large number of receivers. There are various types of FEC mechanisms available for multicasting including pro-active FEC, reactive FEC and layered FEC, none of which are discussed here.

3.1.2.2 Group Reliability and Management

Similar to data reliability, mechanisms for group reliability require feedback from the receiver. It is the responsibility of the transmitter to solicit such feedback and maintain the required information. The level of reliability required of a reliable multicast protocol is determined by needs of the application. The greater the level of required reliability, the greater the amount of information that needs to be maintained by the group members, and hence the greater the AGI. If the application does not care about group membership (0-reliable multicast), then the design of the protocol becomes relatively simpler. On the other hand, if the user's reliability semantics require that all members of the group must remain alive throughout a session (atomic multicast), then the protocol must hold and periodically update all necessary data structures required to ensure this. In such cases, the detection of a failed receiver dooms the integrity of the group.

Efficient group management requires that there be sufficient reliable information about the group members. For applications having a single sender sending to multiple receivers, this state information must be stored and managed at the sender. In order to achieve higher scalability, the receivers may be assigned the task of state management. In case there are multiple senders into a group, the group may be monitored by a globally elected group-coordinator. The control algorithms make use of this state information while the group management policies determines when and how changes in membership (admission of group members, pruning of group members, etc.) are handled.

Some multicast protocols use implicit group management, where sources simply send data to the multicast group address, which receivers can join to receive the data. This model matches the way IP multicast handles group membership and scales well for large groups. Some protocols provide no group management functions, leaving it up to the application. Other protocols may require full control over group membership. In this case, either the membership is static throughout a session, or an explicit group membership protocol ensures reliable joins and leaves.

3.1.2.3 Data Propagation Mechanism

The simplest way to implement a 1-to-N (multicast) conversation would be to make use of N one-to-one (unicast) connections. However, due to obvious inefficiencies, most multicast mechanisms tend to be implemented over broadcast network hardware as was discussed in the previous chapter.

Similarly, multipoint-to-multipoint communication can be made up of N multicast conversations. For instance, an M-by-N application on the Internet may make use of M distinct IP multicast addresses. A better technique would be to make use of a single IP multicast address with some sort of coordination of data flow amongst the participating entities.

3.1.2.4 Assistance from Network Elements

A reliable multicast protocol must involve mechanisms running in end hosts, and must involve routers forwarding multicast packets. It is common for routers to merely forward packets leaving the reliability mechanisms to the transmitters and receivers.

It is also possible for certain network elements to provide a degree of assistance in this area. Routers on the data distribution tree from the transmitter to the receiver may assist in the delivery of data and feedback aggregation and suppression. Since routers can directly influence multicast routing, they have a degree of control over how traffic is routed to various group members.

There may also be a need for certain additional nodes to coordinate a multicast session. The functions of such an entity, which is also known as the coordinator, master or designated receiver may include assisting with data delivery, feedback aggregation, overseeing group membership and operational parameters, keeping track of the session's current state and handling message sequencing via the use of tokens or sequence numbers.

It should be noted that such a entity becomes a single point of congestion and possible failure. A network partitioning may also render the partition without the coordinator inoperative. In order to achieve better reliability, failure detection when there are no messages from the coordinator for a period of time and recovery through the election of a new master needs to be performed. As part of the group reliability mechanism, the master may periodically need to inform the other members of the group of its existence. All the members of the group may in turn be required to reply to that message to indicate to the master that it is alive. This is especially true of most NACK-based receiver reliable protocols.

3.1.2.5 Message Ordering and Priority

Current solutions for local ordering make use of TCP-like sequence numbering of the data. In the case of multipoint-to-multipoint communication, the global ordering is achieved by using a centralised sequencer that grants tokens.

Receivers must ensure that packets with lower token number are delivered to the upper layer prior to those with higher numbers. This may result in certain more important messages being held up due to the non-delivery of messages with lower token numbers. To overcome this problem, a special mechanism called streams may be employed at the receivers whereby every message is assigned a priority and a stream. Within a stream, sequencing is guaranteed, but a stream is completely independent from all other streams; and hence messages can overtake other ones with a different stream number.

3.1.2.6 Scalability

For applications that need to scale to a large group of receivers, the burden of feedback control and error recovery must be distributed among all the members of the group by dividing the global multicast group into separate subgroups with the use of some hierarchy mechanism to collect the responses from the various receivers.

The scalability of applications can also be improved by allowing receivers to maintain and manage state information (using receiver reliable mechanisms as discussed in section 3.1.2.1.2.2), there by relieving the sender of this highly intensive task.

Another approach to achieving scalability is to split the data across multiple multicast groups. There may be a number of different variables based upon which the data can be split. For instance, receivers may join an appropriate group based on the quality of transmission they wish to receive. This may in some cases require fast join and leave functionality from the routers and may also require more forwarding state in them.

3.1.2.7 Late Join and Early Departure Mechanisms

Depending on how valuable the past data is to late joiners, the protocol should require the transmitter to buffer all or part of the data using techniques like multi-level caching. The late joiners may then look at the sequence number of the packet that was currently multicast by the transmitter and send a special request for retransmission of all the packets with sequence numbers lower than the current one.

The question of how long the data needs to be retained by the transmitter also depends on the needs of the application. If the application requires that all the data be available all through the life of the transport connection, then sufficient buffering will be required to satisfy this demand. On the other hand, applications may specify a retention time for the data and once this time expires, the data may be discarded.

The protocol must also contain mechanisms to allow members to indicate their intentions of joining or leaving a multicast group during the data transmission phase.

3.1.2.8 Flow and Congestion Control

Most Internet applications today rely primarily on end-to-end congestion control at the protocol layer. A lot of research [Jaco97] is currently underway in this area.

Rate-based windowed flow control mechanisms may be used in the multicast case to avoid overloading slow receivers and links with low bandwidth. This technique involves having the sender transmit a window full of new packets at the start of a cycle and at the beginning of the next cycle updates the send window and transmits as many new packets as there is room for its send window. The sender must ensure that all the receivers that have sent status messages within a given interval of time, have successfully received the relevant packets before advancing the lower end of its send window. In addition, the sender never transmits more than a full window of packets during a fixed interval, thereby limiting the maximum transmission rate.

Ideally reliable multicast protocols should perform TCP-like congestion control so that they can be TCP friendly. However, this will reduce the rate of transmission to that of the slowest receiver. Since there are always uncorrelated packet losses on different parts of a multicast tree, the transmission rate will eventually decrease to a minimum, i.e., one or two packets per round-trip time. One possible approach in this case is to simply transmit at the rate of one or two packets per round-trip time and open the window very slowly when none of the receivers report any losses. The slow-start algorithm also helps prevent a sender from flooding an already congested network, by detecting possible congestion based on the number of received acknowledgement packets that contain retransmission requests.

Another approach is to have multiple multicast groups and the sender then transmits at different rates to each group. Receivers can choose to join the groups based on the congestion they experience.

3.1.2.9 Quality of Service Mechanisms

There are several mechanisms available for the negotiation and agreement of QoS parameters. A discussion on these mechanisms is beyond the scope of this thesis. More information on QoS mechanisms can be found at [QoS].

3.1.2.10 Security Mechanisms

The simplest way to multicast data securely is to encrypt it before transmission. Issues of security on networks and the Internet in particular pose several interesting research problems which are beyond the scope of this thesis. Readers may refer to [SMuG] for more details.

3.1.3 Reliable Multicast Transport Building Blocks

[Whet99] has defined the concept of a building block to be logical protocol components that result in explicit APIs, which may be used by other building blocks or by protocol clients. These building blocks are generally specified in terms of a set of algorithms and packet formats that implement protocol functional components. A few sample building blocks include NAK-based reliability, congestion control, security, etc.

This thesis makes use of a similar idea and lays out the set of requirements posed by multicast applications in general and proposes a functional decomposition of reliable multicast protocols according to the mechanisms they employ to satisfy the listed requirements. This functional decomposition has also resulted in a number of additional building blocks like late join and early departure mechanisms, QOS mechanisms, etc., which may be used to build multicast transport protocols, all of which have been listed in table 4. In certain cases, a one-to-one mapping of the mechanisms and requirements is always not possible and certain requirements must be satisfied using multiple mechanisms. For instance, hierarchical receiver-oriented reliability is much more scalable if it makes use of special receivers to collect acknowlegments from receivers further down the hierarchy. The table has also omitted the mechanisms necessary to satisfy QoS and security requirements as they are beyond the scope of the thesis.

Requirements			Mechanisms
Type of Reliability	Data	Best-Effort	No reliability mechanism required
		Bounded Latency	Sender Reliable (with time constraints)
		Most Recent	Sender Reliable (with most recent value of parameter constraint)
		Absolute	Sender Reliable
	Group	Connectionless	No reliability mechanism required
		K-Reliable	Transmitter solicited feedback from Receivers
		N-Reliable	Transmitter solicited feedback from Receivers
Multicast Data Distribution	1-by-N		N-unicast (or) broadcast
	M-by-N		M – 1-by-N multicast (or) M-by-N multicast
Ordering	Unordered		No ordering mechanism required
	Local Ordering		Local sequence numbering
	Partial Ordering		Tokens + Local sequence numbering
	Causal Ordering		Tokens + Local sequence numbering
	Total Ordering		Tokens + Local sequence numbering
Scalability			Hierarchical Receiver reliable mechanisms
Late Join			Multi-Level Cache
Flow/Congestion Control			Windowed/Slow start
QOS			(Not discussed in this thesis)
Security			(Not discussed in this thesis)

Table 4: Mapping Multicast Mechanisms to satisfy Requirements

3.2 Review of Multicast Transport Protocols

The following section reviews four well know multicast transport protocols and maps their requirements and mechanisms over our taxonomy in order to demonstrate its completeness. These protocols have been chosen because they map well onto the listed requirements and mechanisms. A description of each protocol has been provided and the list of requirements they claim to satisfy using underlying mechanisms has been tabulated.

3.2.1 Local Group based Multicast Protocol

The Local Group based Multicast Protocol (LGMP) [Hofm97] supports reliable and semi-reliable transfer of both continuous media and data files. The protocol improves scalability and performance by using subgroups (also called Local Groups) for local acknowledgement processing and error recovery, as defined by the Local Group Concept (LGC).

Local Groups are formed by dynamic organisation of receiver sets and are in turn managed by a special receiver called a Group Controller, which handles status responses and coordinates local retransmissions. The selection of appropriate receivers as Group Controllers is based on the current state of the network and of the receivers themselves. This process is not part of the LGMP protocol and is implemented by a separate configuration protocol called Dynamic Configuration Protocol (DCP).

Packet errors are first recovered inside Local Groups using a receiver-initiated approach. The sender performs retransmissions only if no member in a Local Group (including the Group Controller) has a copy of the data. Otherwise, errors will be recovered by local retransmissions.

3.2.1.1 Acknowledgement Scheme

LGC defines three types of acknowledgement packets namely positive acknowledgements (ACKs to release data units from the sender's buffer), negative acknowledgements (NACKs to request for retransmission of missed data units) and semi-negative acknowledgements (SNACKs to indicate that at least one member of the Local Group has correctly received a data unit, but another one is still missing it). Hence a SNACK implies that a data unit has not yet been received correctly, but there is no request for its retransmission.

In case one member of a Local Group does not receive a data unit, local error recovery is first attempted. The GC does not NACK such a data unit. However, the sender is not allowed to release the data unit from its buffer. If the successful receiver that was supposed to perform the local retransmission dies, or leaves the subgroup or group, all the members of the Local Group must have the chance to get the missing data unit from someone else. In the worst case, they will get it directly from the sender. Therefore, the GC does not acknowledge the data unit positively either. Instead, it uses a semi-negative acknowledgement to indicate the status of the data unit.

3.2.1.2 Error Recovery Scheme

LGC performs local retransmissions thereby reducing unnecessary traffic in segments of the network that have no errors. A GC requests a missing data unit from the sender or a higher-level GC only if no member of its subgroup has a copy of that data unit. The GC may either unicast or multicast missing data units within its subgroup. It is also possible for regular receivers to perform retransmissions based on timers carefully set to suppress redundant requests and repetitive retransmissions. The suitability of each approach mainly depends on the application-specific environment and on the characteristics of the underlying network.

LGC defines two different modes of performing local retransmissions: a *load-sensitive mode* and a *delay-sensitive mode*. It is up to the application to choose the appropriate mode according to its requirements. It is also possible for different GCs to operate in different modes.

- **Load-Sensitive Mode:** The aim of this mode is to reduce network load caused by local retransmissions and control messages. Local retransmissions are performed only after the expiration of a predefined time interval. The decision whether to perform retransmissions using unicast or multicast is taken dynamically depending on the current group status. If the number of repair requests for a particular data unit exceeds a certain predefined threshold, then the GC will multicast that data unit, otherwise it will send it directly to the requesting receivers using unicast. However, local retransmissions performed by regular receivers are always multicast to the Local Group.

- **Delay-Sensitive Mode:** This mode is more suitable for time-constrained applications that cannot tolerate delay. Here the GC immediately multicasts the requested data unit to the Local Group on receiving a NACK. If the GC itself is missing a certain data unit, it will multicast a repair request to the Local Group. Any receiver holding a copy of the requested data unit delays the retransmission for a random interval, after which the receiver multicasts the requested data unit to the Local Group. Other receivers that are still waiting to perform the same local retransmission will stop their repair timers on receiving the local retransmission.

3.2.1.3 Congestion Control

LGC separates the signal for indicating congestion from the algorithm for congestion control. It provides mechanisms to detect network congestion based on the status reports of each receiver, but leaves it up to the application to choose the algorithm to deal with congestion.

3.2.1.4 Management of Local Groups

The placement of GCs and the assignment of receivers to appropriate Local Groups are essential for the efficiency of data transfer. As the GCs are also single points of failure, auto-recovery mechanisms are necessary for handling such situations. In addition, due to dynamic changes in memberships of the Local Groups, they must be self-maintaining and self-reconfigurable.

Instead of integrating mechanisms into LGMP to define Local Groups and establish and maintain these logically structured group hierarchies, a new protocol named "Dynamic Configuration Service" (DCS), has been defined. DCS provides all the mechanisms necessary to get the state information but it does not define the parameters for the selection of an appropriate GC, which is a session-level control or QOS management task. As the state information maintained by DCS is distinct and independent of any data-level protocol, it can interact with any other protocol that requires a similar receiver hierarchy.

3.2.2 Reliable Multicast Transport Protocol

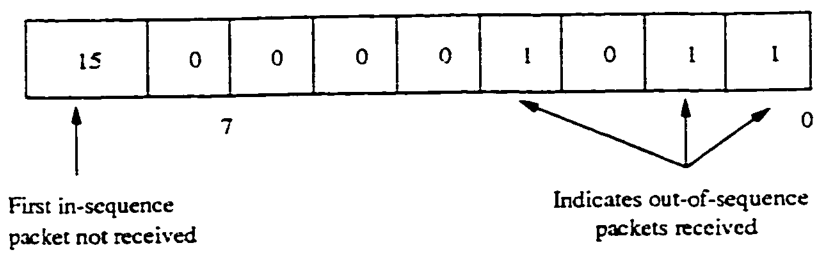
The Reliable Multicast Transport Protocol (RMTP) [Paul97] provides sequenced, lossless delivery of bulk data from one sender to a group of receivers. Reliable data delivery is ensured by selective retransmission of lost packets by the sender in response to retransmission requests from receivers. To avoid throttling the sender with retransmission requests, RMTP uses a tree-based hierarchical approach.

The receivers in a RMTP session are grouped into Local Regions and a Designated Receiver (DR) or ACK Processor (AP) functions as a representative of the local region. The sender multicasts every packet to all the receivers using a "global" multicast tree. Only the DRs send their status to the sender indicating which packets they received and which they did not receive. The receivers in the local regions send their status to their corresponding DR. The DR uses these status messages to perform local retransmissions to the receivers, there by reducing the end-to-end delay. Thus the senders sees only the DRs and the DR sees only the receivers in its local region. By distributing the processing of status messages among the sender and the DRs, the acknowledgement implosion problem is avoided. RMTP also supports multi-level hierarchy, in which case, a DR sends its status to the DR least upstream from itself in the multicast tree and thus, the sender receives only as many status messages as there are DRs in the highest level of the multicast tree. Usually RMTP receivers are grouped into local regions based on their proximity in the network. For example, in IP networks, receivers are grouped into local regions on the basis of the time-to-live (TTL) field in the IP packets. The global multicast tree, rooted at the sender and spanning all the receivers is set up at the network layer (ATM layer in the context of an ATM network). Local multicast trees, which are parts of the global multicast tree are set up between the DRs and their corresponding receivers.

RMTP's design is based on the IP-Multicast philosophy and hence senders do not possess explicit knowledge of the set of receivers. The receivers may join or leave a multicast session without informing the sender. The status information from receivers must reach the sender within a session-specified maximum time period.

3.2.2.1 Session Manager

RMTP sessions are created and controlled by a Session Manager, which is not a part of the protocol. The Session Manager provides all the participants with the associated connection parameters including the send and receive window sizes, the multicast retransmission threshold, data packet size and other parameters that affect protocol performance. It is also responsible for detecting and handling possible exceptional


15
0
0
0
0
1
0
1
1
7
0
First in-sequence packet not received
Indicates out-of-sequence packets received
Figure 14: Logical format of an acknowledgement bitmap

situations such as network partitions and receivers voluntarily or involuntarily leaving the multicast group.

3.2.2.2 Error Recovery

An RMTP sender divides the data to be sent into fixed sized packets and transmits them using the global multicast tree. Every packet is assigned a unique sequence number that defines the overall order before they are multicast to the group.

Designated Receivers learn of missing packets from the ACKs unicast to them by the receivers in their local region. The ACK packets contain the sequence number of the first in-sequence packet not received by the receiver, plus an acknowledgement bitmap indicating which out-of-sequence packets have been successfully received. (Figure 14)

The DRs keeps track of the sequence number of the lowest in-sequence packet that has not been successfully received by a receiver. The data packets not received are added to a retransmission queue, along with the identity of the receiver that requested the retransmission. If the number of receivers requesting retransmission of a packet exceeds a predefined threshold, the packet is re-multicast to all the members of the group; otherwise, the packet is retransmitted via unicast to the receivers who requested the retransmission.

3.2.2.3 Late Joining Receivers

It is possible that some receivers join a session after data transmission has commenced. These late joining receivers must be allowed to catch up with the other receivers. In addition, some receivers may temporarily fall behind due to various reasons such as network congestion or network partitions. RMTP has two features which together allow lagging receivers to receive the missed data.

1. **Immediate Transmission Request:** When a receiver joins a session late, it can find out the packets it has missed earlier by looking at the sequence number of the packet that is presently being multicast by the sender. At that instant, it uses an special ACK packet to request its ACK processor for immediate retransmission of those earlier packets. When an processor receives this special ACK packet from a receiver, it checks the bit vector of the packet and immediately unicasts the missed packets to that receiver.

2. **Two-level Data Cache:** To allow receivers to join an ongoing session at any time and still receive all the data reliably, the senders and designated receivers in RMTP need to buffer the entire file during the session. This allows a receiver to request the retransmission of any transmitted data from its corresponding ACK processor. RMTP uses a two-level caching mechanism whereby the most recent data packets are cached in memory, and the rest are stored on disk.

3.2.2.4 Selection of Acknowledgement Processors

An ACK processor for a receiver is either a sender or designated receiver that processes acknowledgements and retransmits lost packets to the receiver. Receivers and designated receivers periodically send acknowledgements to their ACK processors. Although specific machines are manually chosen to function as designated receivers, the choice of an ACK processor for a given local region is done dynamically.

The designated receiver chosen to be a receiver's ACK processor is the one closest in terms of the number of hops (determined by using TTL's) to the receiver. ACK processors are selected using this mechanism at connection establishment, and when an established ACK processor fails.

3.2.2.5 Flow Control and Congestion Avoidance

RMTP uses a rate-based windowed flow control mechanism to avoid overloading slow receivers and links with low bandwidth. The sender transmits a window full of new packets at the start of a cycle, and at the beginning of the next cycle, it updates the send window and transmits as many new packets as there is room in its send window. The sender ensures that all the designated receivers, that have sent status messages within a given interval of time, have successfully received the relevant packets before advancing the lower end of its send window. In addition, the sender never transmits more than a full window of packets during a fixed interval, thereby limiting the maximum transmission rate.

RMTP implements the slow-start algorithm to prevent a sender from flooding an already congested network. The mechanism detects possible congestion based on the number of received acknowledgement packets that contain retransmission requests.

3.2.3 Multicast Transport Protocol

The Multicast Transport Protocol (MTP) [Arms92] provides sequenced, globally ordered, reliable, rate-controlled, atomic transfer of messages[2] between one or more communicating processes as well as a predefined principal process. It is designed to be used on top of an unreliable, not necessarily sequence-preserving multicast network protocol such as IP multicast.

[2]A MTP message is a concatenation of user data portions of a series of data packets with the last packet in the series carrying a end-of-message indication

Apart from reliable message delivery, MTP uses an ordering and agreement protocol to provide the synchronisation necessary for group members[3] to agree on the order of receipt of all messages and delivery of those messages even in the face of partitions.

MTP-2 [Borm94] is an enhanced version of MTP. It overcomes some of the practical problems associated with using MTP and also adds functionality to it. Self-Organising Multicast (MTP/SO) [Borm97], uses MTP-2 as a basis and adds spontaneous self-organisation of the members of the group into local regions.

The members of an MTP-2[4] group can take one of three different roles: coordinator, sender and receiver. The coordinator provides the message ordering for all members in a group and oversees group membership and operational parameters. Senders send data in messages (a sequence of one or more packets) after obtaining a token from the coordinator. Receivers receive these messages and request the retransmission of lost packets through negative acknowledgements.

The data transfer and retransmissions are based on dividing time into heartbeat intervals (measured in microseconds). After the initial transmission of a packet, receivers have a limited retention time (measured in heartbeats) to request its retransmission. After that time, senders are no longer obliged to honour NAK's, allowing the producer to discard its copies of the data sent.

Before sending messages, a sender obtains a token from the coordinator that contains a global message sequence number. Receivers are responsible for delivering messages in the correct (global) order to their applications. The token allows the sender to transmit a limited number of packets in a heartbeat interval, thereby achieving rate control.

[3] In order to maintain consistency throughout the document, the terms group, coordinator, sender and receiver have been substituted for the terms web, master, producer and consumer.

[4] As several limitations of MTP have been overcome in MTP-2, the rest of the discussion deals with MTP-2, unless otherwise specified

3.2.3.1 Coordinator Control

The concept of coordinator is very essential to the proper functioning of the MTP protocol. The coordinator is the principal member of the group and is responsible for instantiating and controlling the behaviour of the group, including its membership and performance.

The coordinator handles group membership by admitting or rejecting members wishing to join the group via a join request packet. A member may not be admitted into the group in cases where it specifies service parameters that are in conflict with those established by the coordinator. The coordinator has the capability to regularly check the membership status of the group by querying individual group members. A member may leave an on-going session by informing the coordinator of its intention to quit the group.

The coordinator is also responsible for issuing tokens, which are necessary for transmission into the group. Because MTP is a many-to-many protocol wherein any number of senders may transmit to the group, the tokens also ensure total ordering of messages.

3.2.3.2 Global Ordering

The coordinator assigns a global sequence number (also called a message sequence number) to each message, which it sends in a token confirm packet as a response to a token request. Senders will send this sequence number in every data packet belonging to the message. To order packets within a message, senders assign a packet sequence number on a per-message basis. It is the responsibility of the receivers to deliver messages in the correct order to the applications, if sequenced delivery has been specified for a message.

Senders return the token to the coordinator implicitly with the final packet of the message once they have finished transmitting all their data.

3.2.3.3 Error Detection and Recovery

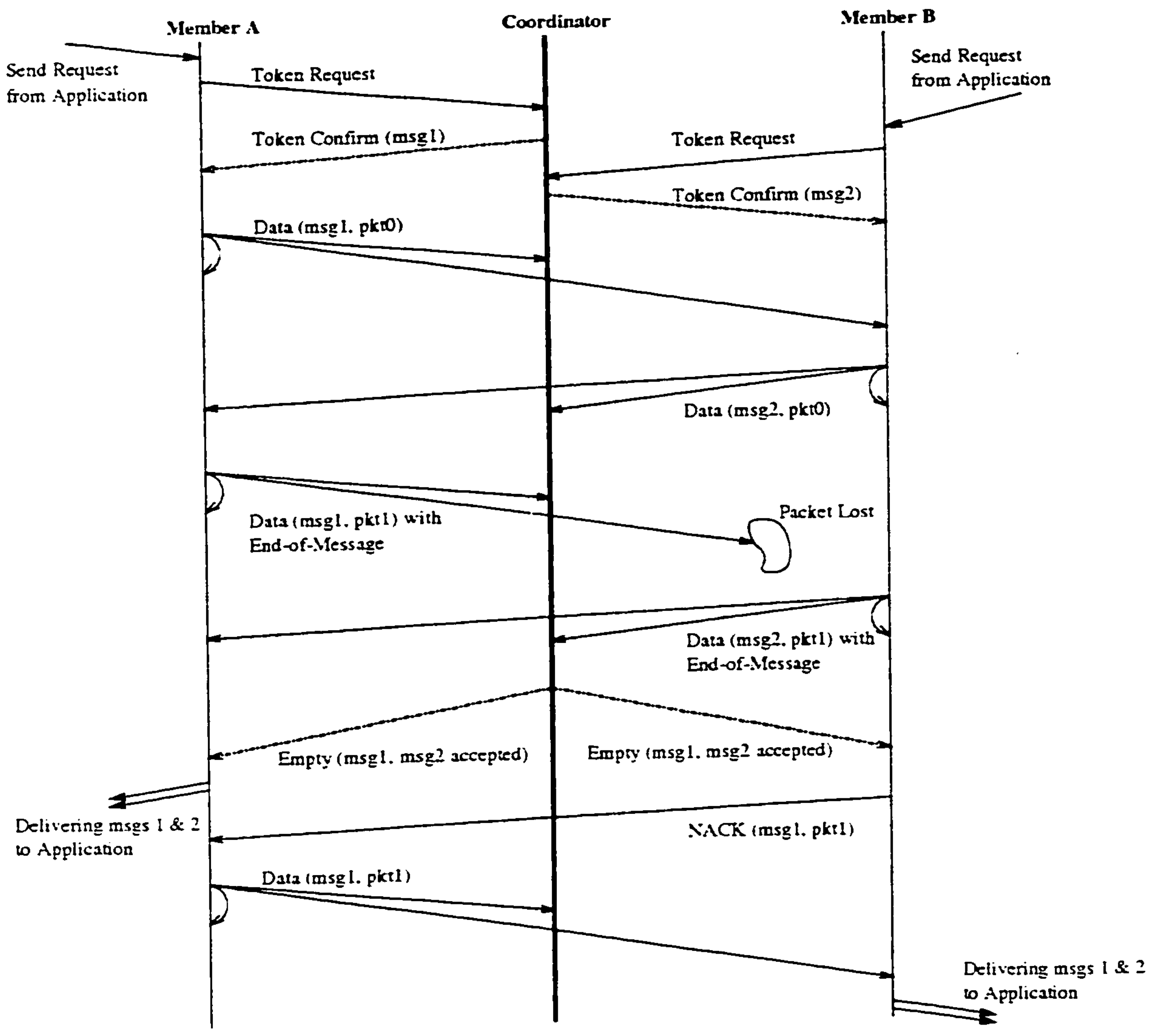

Member A
Coordinator
Member B
Send Request from Application
Token Request
Send Request from Application
Token Confirm (msg1)
Token Request
Token Confirm (msg2)
Data (msg1, pkt0)
Data (msg2, pkt0)
Data (msg1, pkt1) with End-of-Message
Packet Lost
Data (msg2, pkt1) with End-of-Message
Empty (msg1, msg2 accepted)
Empty (msg1, msg2 accepted)
Delivering msgs 1 & 2 to Application
NACK (msg1, pkt1)
Data (msg1, pkt1)
Delivering msgs 1 & 2 to Application

Figure 15: Operation of MTP-2 Protocol

When a receiver detects missing packet sequence numbers, they unicast NACKs to the sender. Retransmissions are then multicast to the group (Refer Figure 15).

To limit buffering at the sender, receivers have a limited time to request retransmission of data packets. This time is called retention time and is measured in heartbeats. After this time, senders are no longer obliged to honour NAKs, allowing them to discard their copies of the data sent.

3.2.3.4 Atomic Message Transfer

MTP-2 ensures that all members agree on which messages are accepted by assigning a status to each message. The coordinator maintains a message acceptance record containing the status of the most recent 12 messages. Messages that are in progress are marked "pending", messages that have been correctly received by the coordinator are marked "accepted", and messaged that have not been correctly received are marked "rejected". The message acceptance record is transmitted in every packet sent by the coordinator.

As soon as the sender notices one of its messages to be accepted, it sends an acknowledgement of successful transmission to its application. Such an acknowledgement does not guarantee that every receiver has received the message but that at least the coordinator was able to receive it correctly. If a specific receiver does not completely receive a message (even after requesting retransmission), that is accepted by the coordinator, it will signal this as an unsuccessful reception error to its application. It may also be an indication that such a receiver may be failing.

Receivers do not deliver pending or rejected messages to the application. A message marked as rejected was not completely received (even after repeated retransmissions) by the coordinator. Normally, every receiver will drop such messages and the sender of the message will indicate an unsuccessful transmission error to its application.

It is obvious that atomicity increases message latency because applications need to wait for the acceptance state of messages to propagate from the coordinator before they can act on them. MTP-2 provides the facility to turn off atomicity on a per-message basis, should the application not require it.

3.2.3.5 Coordinator Loss and Migration

MTP-2 provides improved reliability by providing a mechanism for continuing with the session in case the coordinator fails or is partitioned from a set of receivers. All members can detect the loss of the coordinator when they do not receive any packets or new parameter values from it. The suspected loss of the coordinator is confirmed by the receivers sending it a special packet. If the coordinator does not respond, the members assume that it has failed and elect a new coordinator, which accumulates information about the status of all active messages, as well as recently requested transport parameters from all responding members.

Coordinator migration is useful when the host with the running coordinator or its network connection is overloaded or in the case when the member to which the coordinator is associated wants to leave the session. The procedure adopted is similar to the coordinator loss recovery procedure.

3.2.3.6 Dynamic Parameter Adjustment

The transmission time is divided into equally spaced intervals called heartbeats. When no data is being sent, the master multicasts a special packet (called an empty packet) per heartbeat. To reduce the impact of this overhead, the heartbeat can be slowed when senders are quiescent. The amount of time that senders may retain data may also be adjusted.

Also if frequent loss is encountered, higher parameter values may increase the likelihood of successful (re)transmission. Only the coordinator of the group can modify the heartbeat, retention and window parameters.

3.2.3.7 Priority and Streams

MTP-2 provides priority assignment for token requests and the coordinator processes these requests in the order of priority.

It also provides for sequencing of high priority messages as it would sequence normal messages using a concept called "streams". In MTP-2, every message has a priority and a related stream. Within a stream, sequencing is guaranteed, but a stream is completely independent of all other streams; therefore messages in a stream can overtake others that are in a different stream.

3.2.3.8 Self-Organisation

The MTP/SO protocol was developed to handle the NACK-implosion problem, by introducing hierarchy to MTP-2. MTP/SO introduces the concept of regional repeaters. Receivers multicast NACKs with a local scope[5], before multicasting them with a larger scope. Repeaters that have previously received the requested data retransmit locally after receiving a local NACK. Repeaters that don't have the data just relay the NACK to the next higher level of hierarchy, up to the whole group, where, finally, the sender itself replies with a copy of the data.

3.2.4 Xpress Transport Protocol

The Xpress Transport Protocol (XTP) is a high-speed transport protocol designed to meet the needs of various applications ranging from real-time embedded systems to multimedia distribution to applications distributed over a wide-area network. It defines a set of orthogonal mechanisms necessary for delivering user data from one end-system to one or more end-systems but avoids implementing specific pre-established policies. It provides a multitude of independent features including flow control, rate control, error control, delivery priority, unicasting/multicasting, various data delivery semantics, parametric address and traffic specification from which the user can construct a protocol well-suited to the application at hand.

[5]The scoping can be done with IPv4 TTL thresholds or by using IPv6 administrative scoping, but the exact number and extent of scopes is a global parameter of the group

XTP makes use of well-defined packet structures, containing user data or control information, which are exchanged by the end-systems for purpose of data transfer. The control information is used to provide the requested level of correctness via error control algorithms and maintenance of state information. Efficient data transfer and the desired quality of service can be achieved using flow and rate control algorithms, certain protocol modes and traffic shaping information. All of XTP's packet types use a common header structure, which contains the information necessary to steer the packet's payload to the appropriate destination.

3.2.4.1 Contexts and Associations

A "context" is an XTP data structure that holds relevant state information about an active endpoint at a XTP host. Every active conversation in XTP requires a context which must be instantiated before sending or receiving XTP packets. Each context (unicast communication) manages a full-duplex data stream[6] and is also capable of sending and receiving control information. The operation of a context is controlled by bit-flags in the packet header.

An XTP user communicates with one or more XTP users by establishing an "association", which is an aggregate of all the active contexts and data streams between them. To establish an association, the initiating host sends a association-establishment packet to a destination host, which in turn creates a corresponding endpoint for the association. Such an association is a one-to-one or unicast association. XTP also provides for one-to-many or multicast associations in which case there are more than two endpoints. Data flow is full-duplex only in unicast associations. In case of multicast associations, the data flows only in the one-to-many direction. The protocol also provides a variety of mechanisms for closing a connection once all the data transfer has been completed. Figure 16 illustrates contexts and associations

[6]In case of multicast communication, the data stream is simplex.

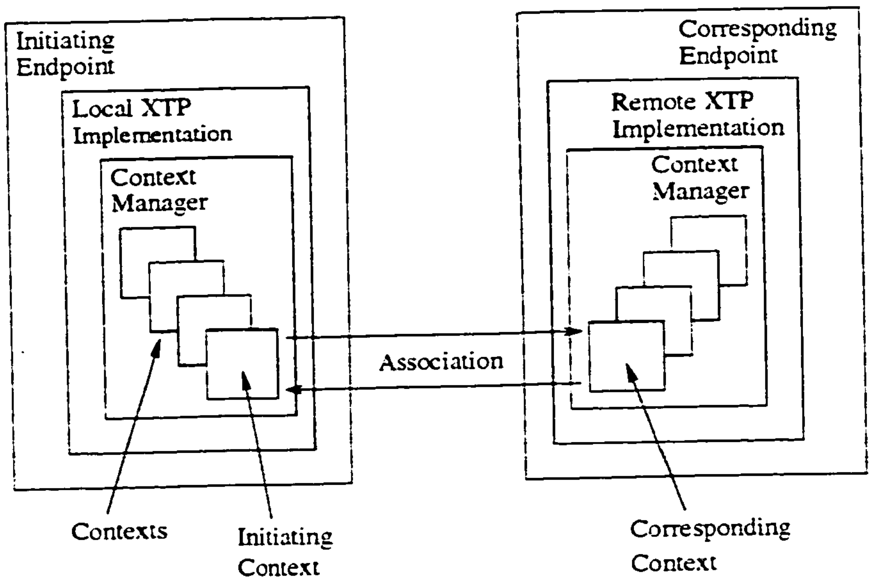


Figure 16: XTP Communication Model

in the XTP communication model.

Only the association-establishment packet carries explicit addressing information. All other packets carry a unique key in their header, which allows the packet to be mapped to the appropriate context at the receiving host.

XTP requires that there be a listening context in the unicast case (or a set of listening contexts in the multicast case) to which a transmitting context can send a packet for association establishment. If there is no endpoint waiting to receive such a packet, it is a protocol error. Such behaviour makes XTP an "asymmetric" protocol.

3.2.4.2 Error Control Mechanisms

XTP's error control mechanisms make use of positive and, when appropriate, negative acknowledgement to effect retransmission of missing or damaged data packets. The retransmission mechanism may either be Go-Back-N or selective-repeat. Missing data is indicated using control messages and this prevents spurious and redundant retransmissions, which may otherwise lead to congestion. There also exist mechanisms for quick-acting error notification.

3.2.4.3 Flow Control and Rate Control Mechanisms

Flow control is a set of techniques that enable a data source to match its transmission rate to the currently available service rate at the receiver and in the network. It links the transmission rate afforded by the transmitter to the buffer space available for the connection at the receiver. Flow control is one of the mechanisms for congestion control.

Rate control on the other hand, is a producer/consumer concept, which deals with controlling the rate at which the transmitting transport entity is allowed to submit data to the underlying data delivery service. It is desirable since the receiver may drop packets not because it has run out of buffer space (flow control problem), but because it cannot process data as quickly as the transmitter sends it.

XTP provides orthogonal mechanisms for shaping rate control and flow control. The flow control is based on 64-bit sequence numbers and a 64-bit sliding window.

3.2.4.4 Support for Multicasting

An important and distinguishing feature of XTP is the provision of mechanisms for transport layer multicasting. XTP multicast is not an attachment to the unicast functionality, rather, each mechanism used for unicast communication is available for multicast as well. It provides a wide range of options from the UDP style best-effort multicast delivery to ordered and reliable multicast delivery. A detailed description of XTP multicast can be found in Appendix A of this thesis.

3.2.4.5 Data Delivery Service Independence

With increasing use of switched networks in place of routed networks, a traditional network layer service may not be appropriate in every instance. Though XTP is a transport protocol, it can be employed over just about any underlying data delivery service including MAC, IP, AAL5 etc. XTP employs parametric addressing, allowing packets to be addressed with any one of several standard addressing formats.

3.3 Summary

Table 5 provides a summary of the various features exhibited by the various multicast protocols discussed in the previous section. It is clear that only XTP provides a complete set of mechanisms to satisfy all the listed requirements.

Protocol	Data Reliability	Group Reliability / Management	Data Propagation	Ordering	Repair Request	Retransmission Mechanism	Flow / Congestion Control
LGMP	Receiver reliable, Heirarchical receiver oriented, using NACKs / SNACKs	- / Explicit	Multicast	Local	Multicast / Unicast	Multicast / Unicast	- / -
MTP	Receiver reliable, sender oriented, using NACKs	- / Explicit	Multicast	Total	Unicast	Multicast	Rate / -
RMTP	Receiver reliable, Heirarchical receiver oriented using ACKs	- / Implicit	Multicast	Local	Unicast	Multicast / Unicast	Window / -
XTP	Sender reliable using ACKs / NACKs	0 to N-reliable / Explicit	Multicast / Unicast	Local	Unicast	Multicast / Unicast	Window / Rate

Table 5: Summary of functionality exhibited by various multicast transport protocols

Chapter 4

Multi-sender Communication using XTP

4.1 Introduction

The previous chapter described two main classes of multicast applications, namely non-interactive point-to-multipoint data distribution applications and multipoint-to-multipoint interactive applications. It is clear that the reliability requirements for these two classes of applications are quite different. RMTP and LGMP are examples of multicast transport protocols that provide support for the former set of applications. They are receiver-reliable and make use of hierarchical mechanisms to achieve scalability. MTP and MTP-2 on the other hand support multipoint-to-multipoint applications but the absence of any hierarchy limit their scalability. MTP/SO overcomes this limitation by introducing mechanisms for hierarchy. The MTP family of protocols are also NAK-based protocols.

Multicasting has always been an integral part of XTP since the early stages of development of the protocol. Revision 3.6 of the specification [XTP3.6] required receivers to multicast their control responses to the group and indicated packet errors using NACKs. An appendix to the specification described several new heuristics including the bucket algorithm, damping, slotting and cloning, as mechanisms to improve the efficiency and effectiveness of XTP's group communication support. Multicasting of responses to the group was not a very advantageous mechanism and caused problems in widely-distributed networks. Moreover the use of NACKs for signalling errors prevented fully-reliable data transfer. Revision 4.0 of the specification [XTP4.0] substantially changed the multicast procedures. The recommendation to suppress response packets was removed, and receivers unicast control packets back to the transmitter. Mechanisms were introduced to uniquely identify participants in an association. However, these mechanisms involved potentially inefficient mappings thereby substantially reducing the protocol performance. These and other issues were corrected in the addendum to the 4.0 specification [Atwo96a].

XTP, like other sender-reliable multicast protocols, requires the sender to keep track of the state of all the receivers using responses gathered from the active receiver group. As already mentioned in section 3.1.1.1.2 on page 29, such a technique does not scale to large groups due to acknowledgement implosion. [Atwo96b] and [Hofm96] suggest two independent ways of dealing with this. Both the proposals make use of hierarchy as a means of improving scalability. The difference between the two lies in the way they handle distribution of data to the members of the group and the way they collect responses from the receivers.

4.2 Previously Proposed Extensions to XTP Multicast

The multicast mechanisms in XTP provide a reliable 1-by-N communication facility. An appendix to the XTP specification [XTP4.0b] describes a set of mechanisms that can be used to build a multipoint-to-multipoint service. This M-by-N communication facility, in which application data flows from M different data sources to N data sinks, makes extensive use of XTP's standard 1-by-N multicast service. An M-by-N multicast communication service has been developed by adding a few extensions to XTP's basic multicast facility. These extensions make use of two techniques called "concentration" and "cloning" to achieve the required functionality.

Concentration is defined as the inverse of multicasting, where arbitrary messages are reliably transmitted from a set of hosts in a multicast group to a single host. If multiple data streams are concentrated into one receiving host, this host requires M contexts to handle data from M sources. XTP implements each concentration data stream as a new unicast association. Cloning is a technique used to improve the efficiency of concentration. In case a large number of concentration channels are needed, there may be an advantage of creating additional contexts (also called pseudo-contexts) automatically instead of using explicit association setup procedures. The simplest method is to implement a persistent listen operation at the concentrator node, which clones a sequence of active contexts in response to incoming FIRST packets.

The existing proposals for the M-by-N service attempt to provide reliable, ordered atomic transfer of messages. A set of M communicants concurrently transmit messages to each other (symmetric group communication), and the messages are reliably delivered to each member in the group with a mutually consistent ordering at N receiving sites. Figure 17 shows the communication setup for an M-by-N service. An application entity at the "sequencer node" sets up a 1-by-N reliable multicast connection with the set of receivers in the multicast group. When a receiver accepts the FIRST packet sent to establish the reliable multicast association, it sends back a unicast FIRST packet to the sequencer node. Using cloning, the transmitting context at the sequencer node establishes a reverse channel with each group member. Thus

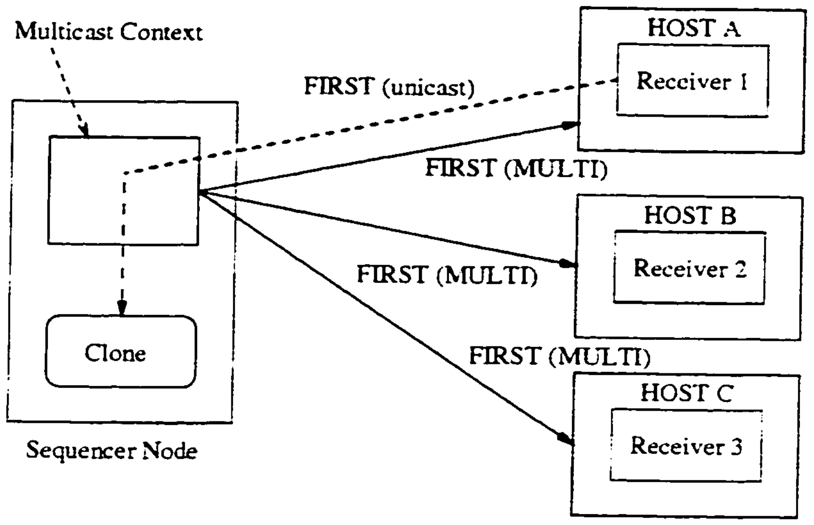


Figure 17: M-by-N Connection Setup

each group member has a reliable XTP unicast connection to the sequencer node, which in turn has a reliable multicast connection to the set of receivers. It should be noted that this mechanism is similar to the one described in page 22 for the ATM multicast server model.

When a group member wants to send a message, it does so asynchronously by unicasting it to the sequencer node (labelled 1 in figure 18), which sends out the message on the multicast connection (labelled 3). The relaying of messages is performed by a transport layer "bridge", which multiplexes the data from a set of receiving contexts into a single (multicast) transmitting context (labelled 2). The bridge is also responsible for preserving message boundaries when forwarding data from the back-channels into the outgoing multicast channel. Flow and rate control can be used on the incoming back-channels to throttle members. The XTP associations ensure reliable delivery and a mutually consistent message ordering of the global message stream at each group member. The XTP group management facility helps the sequencer node detect failures using information from the active group members.

Despite all XTP's strengths, the M-by-N communication mechanisms in XTP as defined the appendix of the 4.0 specification are neither graceful nor efficient. The

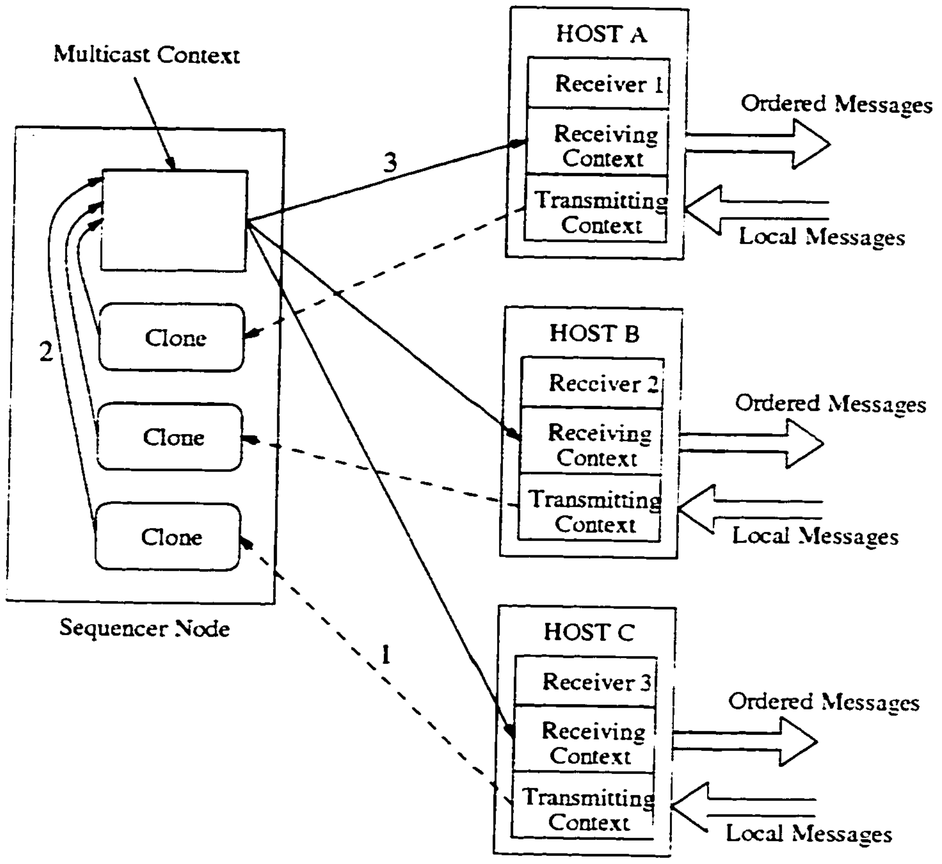


Figure 18: M-by-N Reliable Ordered Multicast

rest of this section takes a closer look at the deficiencies and existing alternatives. IP Multicasting is inherently M-by-N in nature and any number of transmitters may send data to a single IP Multicast address on which any number of receivers may be listening. Unlike other multicast transport protocols, XTP does not make use of this underlying capability. Instead it struggles to achieve similar functionality using the sequencer node, which results in a single point of congestion and failure. The scalability of the system is also greatly reduced because of the sequencer.

One solution to the problems mentioned above is the use of M individual 1-by-N multicast associations. This requires M distinct multicast addresses (one for each sender) on which the N receivers must listen. Such a facility will also require the services from a higher layer to manage the M different multicast addresses. Though this technique solves a few problems described above, it makes inefficient use of the multicast address space. Moreover, this solution requires the use of N contexts for each transmitter for a total of MN contexts.

4.3 A New Proposal for Multi-sender Communication

It is now well understood that the total reliability requirement dictates senders to have complete knowledge of the group (sender-reliable multicast). It has also been repeatedly mentioned that maximum scalability can be achieved when the members of a multicast group are organised into a hierarchy. With these two issues in mind, there is ongoing work [Hanna] to incorporate hierarchy into XTP multicast.

Most M-by-N multicast applications that exist today do not pose heavy scalability requirements. This is inherently due to the fact that applications such as video conferencing typically have few group members (less than 10). In such cases hierarchy is not necessary. It should be recognised though that in future there may exist M-by-N applications in which M is considerably less than N. Examples of such applications include a video conference that has only 4 or 5 members who are active participants while there may be a 1000 other members who are passive listeners.

There is general agreement amongst the members of the IETF that a single reliable multicast protocol or framework is not likely to meet the needs of all Internet applications [Mank98]. However, there is little understanding as to the exact relationship between application-specific requirements and more generic underlying reliable multicast mechanisms.

The International Standards Organisation (ISO) and the International Electrotechnical Commission (IEC), on seeing the lack of a "one-size fits all" solution for reliable multicast, have defined a new transport service interface named Enhanced

Communications Transport Service (ECTS) [JTC1/SC6]. This draft standard attempts to provide a uniform and universal service interface between transport protocols and applications that require support for powerful multimedia group communication. ECTS has been designed to allow it to work on top of any of the well-known unicast and multicast Internet transport protocols such as TCP, RTP, UDP and MTP. A companion protocol to ECTS has also been developed, called Enhanced Communications Transport Protocol (ETCP) [JTC1/SC6], which makes extensive use of the multicast capabilities of IPv4 and IPv6 and relies on RSVP for QOS provisioning through resource reservation.

The ideas resulting from various global standardisation efforts as described above and the availability of a mechanism-rich transport protocol such as XTP motivates us to build an underlying transport service infrastructure to support a wide range of application-specific requirements, most of which have already been listed in the taxonomy. In particular we would like to address the underlying problems with the M-by-N mechanisms in XTP by defining a new multipoint-to-multipoint transport facility. In the next section, we develop a generalised service interface, which will be able to reside on top of the modified transport protocol and perform functions such as packet ordering amongst multiple senders.

The asymmetric nature of XTP's association formation necessitates formation of a multicast association with a process sending FIRST packets to a group of listening contexts. On the Internet, such a group of listening contexts is identified by a multicast group address. The mechanisms necessary to pass this addressing information to the group are not discussed here. It is assumed that a session management application such as the Session Directory Tool [SDR] is capable of conveying the necessary information.

Figure 19 shows the architecture for our new proposal. For ease of explanation, we make use of an example in which 4 processes (labelled A, B, C and D in the

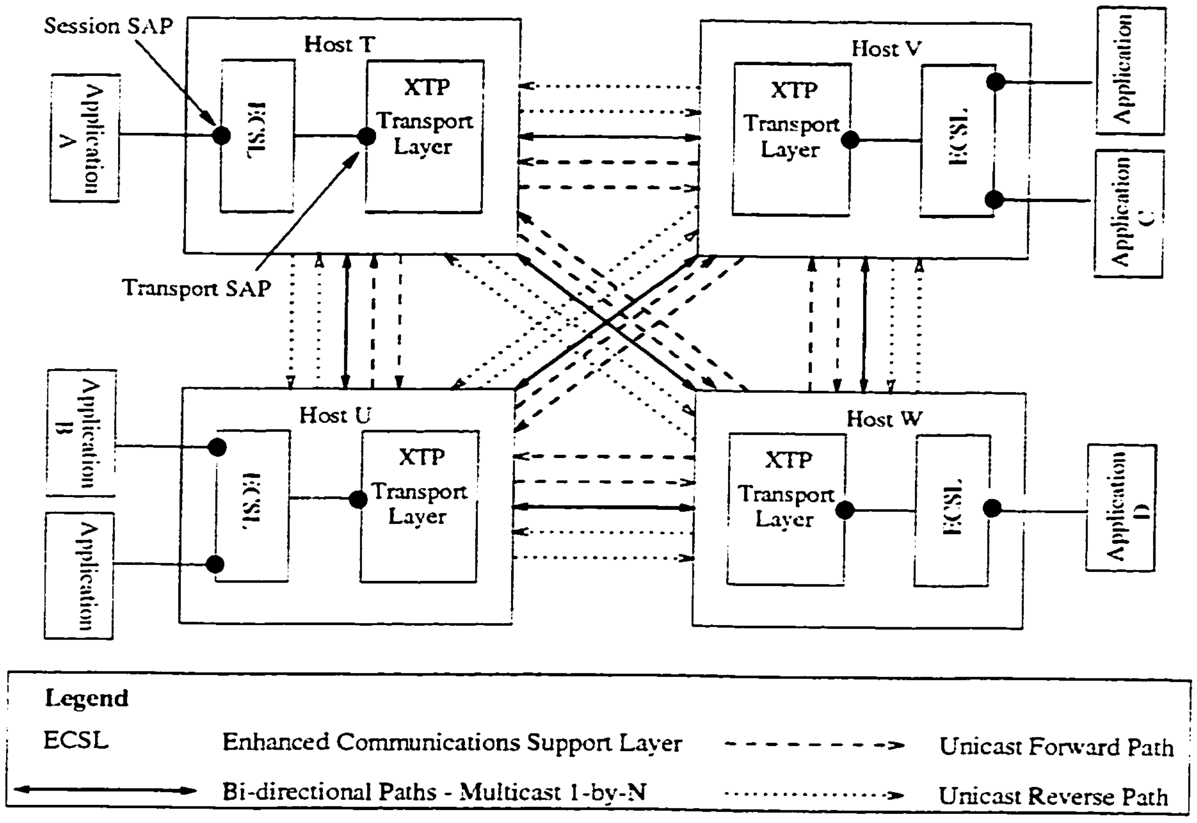


Figure 19: Architecture of the proposed M-by-N communication model

architecture) at hosts T, U, V and W respectively, wish to participate in a multi-party conversation. A separate implementation of XTP's transport layer can be found in each host in our example. Above this layer is our proposed Enhanced Communications Support Layer (ECSL), a non-transport functional layer, which deals with issues such as data ordering and token management. To further simplify our explanation, we assume that only one process at each host wishes to be a part of the multi-party conversation. It should be noted that the ECSL's multiplexing capability could allow more than one process to be part of the M-by-N group at a particular host (See Host U in figure 19). We also make an assumption that the process A at host T begins the session, by sending out FIRST packets to the 3 other waiting contexts at hosts U, V and W respectively (Refer (a) in figure 20). On account of this, T is also assumed to be the master of the M-by-N multicast session, henceforth called the "association" master context.

The association master is the process at a given host that first transmits XTP FIRST packets for a given M-by-N group. It is also responsible for distributing transmit tokens and coordinating the communication between the entities. A more robust protocol implementation may require master selection via some sort of distributed consensus protocol. In such a case, if the host with the association master crashes or is partitioned away, a re-election takes place to find a new master. A discussion on the various distributed consensus algorithms to deal with master re-election is beyond the scope of this thesis.

In the XTP multicast specification, the restriction of data flow from a sender to the multicast receivers is indicated by setting the RCLOSE bit in the XTP header. For data to flow in the reverse direction, separate unicast paths need to be set up. In contrast, for our model, we clear the RCLOSE bit in the FIRST packet, thereby allowing data on the reverse path of the full duplex channel. The sender makes use of the local key for the multicast group K_{T_g} for multicast data transmission. The processes at U, V and W each have a separate listening context, henceforth referred to as "host" master contexts, which wait for the arrival of the FIRST packet from the association master context at host T.

When a host master receives the FIRST packet, it performs a full context lookup. Unlike the cases of unicast and 1-by-N multicast[1] where a full context lookup will fail if the address is in use, it does not fail in the M-by-N multicast case, as it is possible to have more than one sender into a group. Another difference between 1-by-N multicast and M-by-N multicast arises due to the fact that in the former, there may be more than one listening context at a host and hence the search for listening contexts must continue until all are found. On the other hand, in the later case, the search for listening contexts stops when one (host master context in this

[1]Refer A.4.1 for details on first packet matching for 1-by-N multicast.

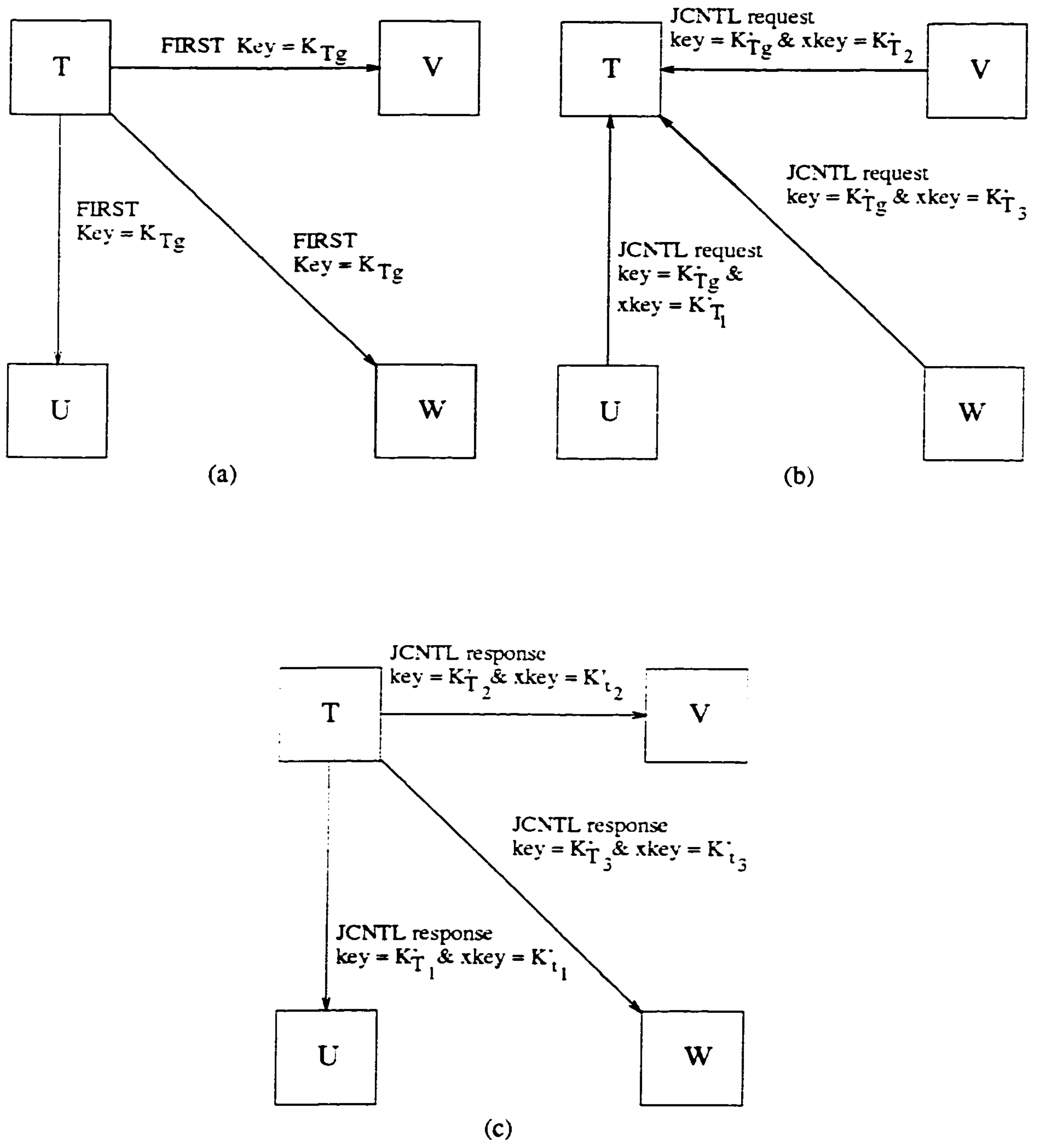

T
V
U
W
FIRST Key = K_Tg
FIRST Key = K_Tg
FIRST Key = K_Tg
(a)
JCNTL request key = K'_Tg & xkey = K'_T2
JCNTL request key = K'_Tg & xkey = K'_T3
JCNTL request key = K'_Tg & xkey = K'_T1
(b)
JCNTL response key = K'_T2 & xkey = K'_t2
JCNTL response key = K'_T3 & xkey = K'_t3
JCNTL response key = K'_T1 & xkey = K'_t1
(c)

Figure 20: Keys exchanged by host T when it wishes to transmit

case) is found. As mentioned earlier, the multiplexing capability of the higher layer would allow multiple processes on a single host to receive the data.

The host master then creates a "subordinate" context, which will handle all further unicast transmissions towards the node that transmitted the FIRST packet (T in this case). Once the FIRST packet satisfies the acceptance criteria, each subordinate context replies with a JCNTL request packet back to the process at host T (Refer (b) in figure 20). Hence U will send a JCNTL request with key = K'_{T_g} and xkey = K'_{T_1} and V will send a JCNTL request with key = K'_{T_g} and xkey = K'_{T_2} and so on. It should be understood that any reference to the host implies the XTP implementation at the host unless otherwise indicated.

Now the association master context at T will create subordinate contexts to handle the incoming unicast JCNTL request packets. Then each subordinate will in turn respond with a JCNTL response packet with key = K'_{T_1} (towards host U) and xkey = K'_{t_1} (Refer (c) in figure 20) and so on. Once all the required packet exchanges have been completed, host T will have one association master context (which will also play the role of the host master context) and three subordinate contexts (one each for the unicast associations to the respective subordinate contexts at hosts U, V and W). Similarly each receiving host will now have one host master context and one subordinate context.

Figure 21 shows the data paths established at the transport layer of our architecture after the above mentioned packet exchanges. If the process at T wishes to transmit data to all the group members, it will use the local key of the multicast group (K_{T_g}). If the transmitter at host T sent packets to a specific receiver (say at Host U), it will make use of the key K'_{T_1} . Likewise, if a member of the group (say at Host U) wanted to send control packets back to the transmitter at T, it will use either the local key of the multicast group as a return key (K'_{T_g}) or the key assigned by the transmitter to uniquely identify receiver U(K'_{t_1}). Note that a member at U may also

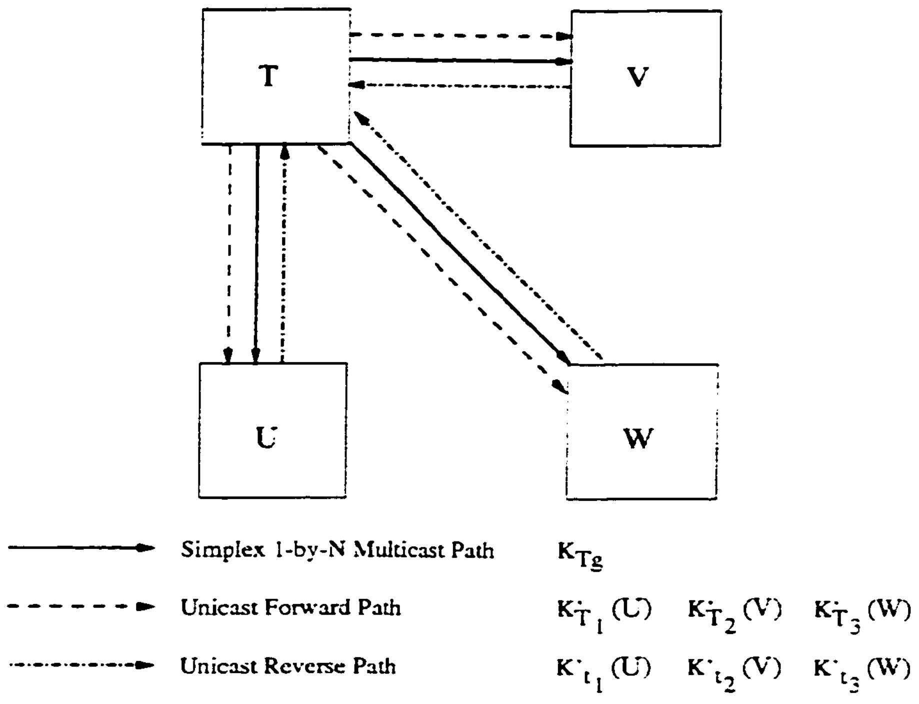

T
V
U
W
Simplex 1-by-N Multicast Path
K_{T_g}
Unicast Forward Path
K'_{T_1}(U) K'_{T_2}(V) K'_{T_3}(W)
Unicast Reverse Path
K'_{t_1}(U) K'_{t_2}(V) K'_{t_3}(W)

Figure 21: Data-Flow at the Transport Layer when process at T transmits

send data back on the reverse path from T because the RCLOSE bit is cleared in the XTP header.

The mechanism we have described so far is similar to the 1-by-N multicast mechanism. Where our model differs is when another process (say at Host U) wishes to begin sending into the group. In such a case, it will first acquire a transmit token from the association master at host T (see section 4.4 for details) and will send out a FIRST packet to the multicast group with key = K_{U_g} with its RCLOSE bit cleared (Refer (a) in figure 22).

As mentioned earlier, each of the participating hosts (T, U, V and W) has a host master context listening for FIRST packets sent to that particular multicast group address. Now each host master context will create subordinate contexts again, which in turn respond with a JCNTL request packet (Refer (b) in figure 22). Thus the host at T will reply with key = K'_{U_g} (or K'_{T_1}, on the unicast forward path set up by the process at T) and xkey = K'_{U_1} (or K'_{t_1}, on the unicast reverse path set up by

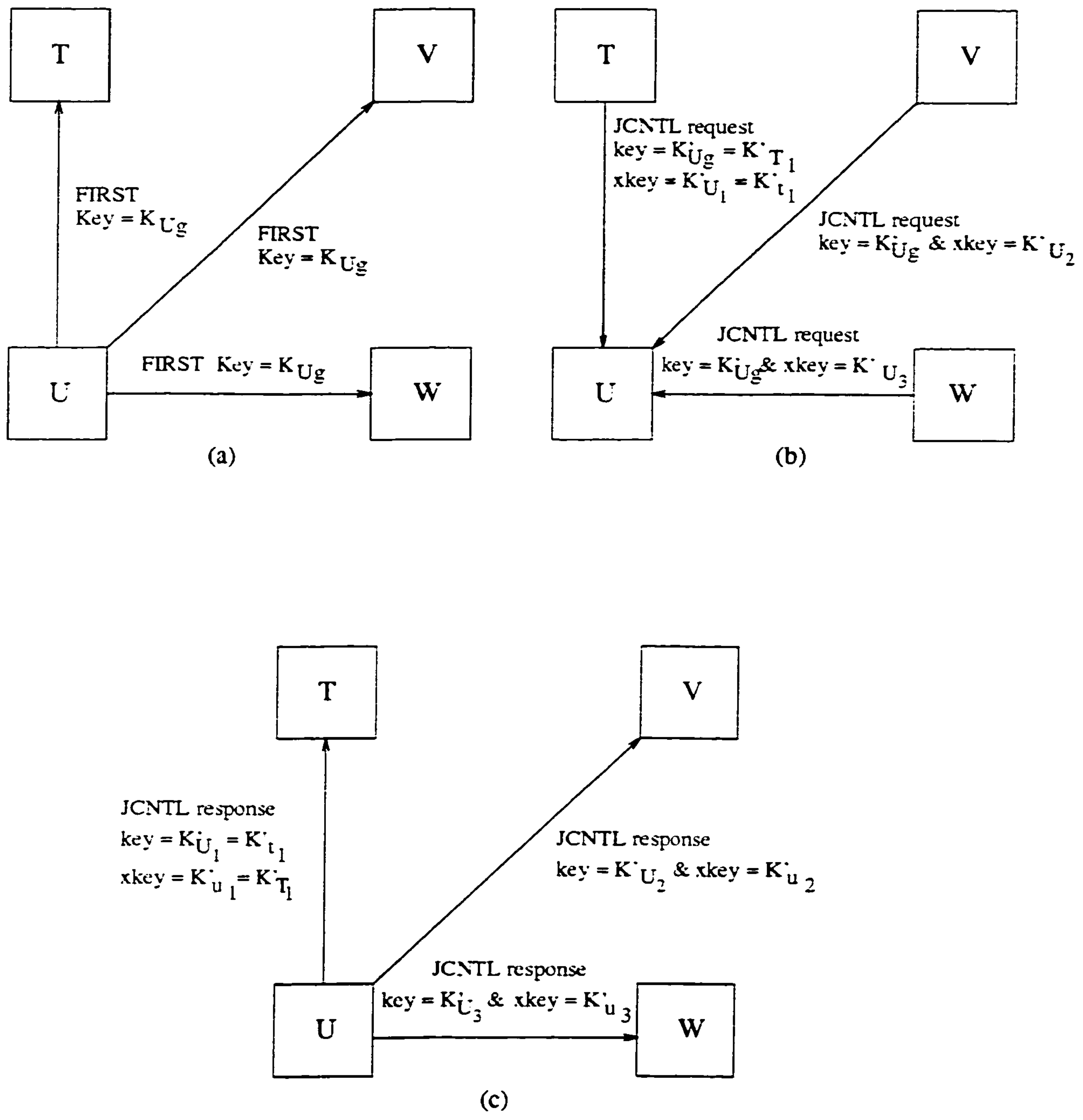

T
V
U
W
FIRST Key = K Ug
FIRST Key = K Ug
FIRST Key = K Ug
(a)
T
V
U
W
JCNTL request key = K'Ug = K'T1 xkey = K'U1 = K't1
JCNTL request key = K'Ug & xkey = K'U2
JCNTL request key = K'Ug & xkey = K'U3
(b)
T
V
U
W
JCNTL response key = K'U1 = K't1 xkey = K'u1 = K'T1
JCNTL response key = K'U2 & xkey = K'u2
JCNTL response key = K'U3 & xkey = K'u3
(c)

Figure 22: Keys exchanged by host U when it wishes to transmit

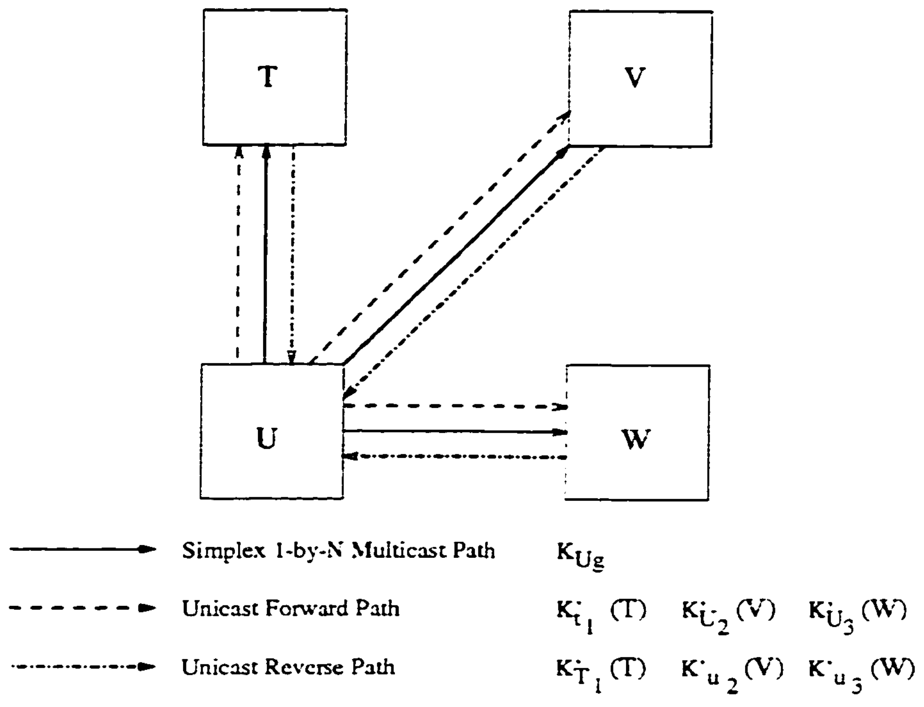

T
V
U
W
Simplex 1-by-N Multicast Path
K_{Ug}
Unicast Forward Path
K'_{t_1} (T) K'_{U_2} (V) K'_{U_3} (W)
Unicast Reverse Path
K'_{T_1} (T) K'_{u_2} (V) K'_{u_3} (W)

Figure 23: Data-Flow at the Transport Layer when process at U transmits

the process at T). The host at U will in turn respond with a JCNTL response packet (Refer (c) in figure 22), which will contain key = K'_{U_1} (or K'_{t_1}, on the unicast reverse path set up by the process at T) and xkey = K'_{u_1} (or K'_{T_1}, on the unicast forward path set up by the process at T).

Now if the process at U wishes to transmit data to all the group members, it will use the local key of the multicast group (K_{U_g}). If the transmitter at U wished to send packets to a specific receiver (say at Host T), it will use key = K'_{t_1}. If a member of the group (say at Host T) wants to send packets back to the transmitter at U, it will use K'_{T_1} (Refer figure 23). Similarly, the packet exchanges required in case a process at host V and host W wish to transmit data are shown in figures 24 and 26 and the corresponding data flows at the transport layer are shown in figures 25 and 27 respectively. Tables 6 and 7 list the keys used by the respective transmitters in the above example.

Most applications are considerably simplified when the messages exchanged

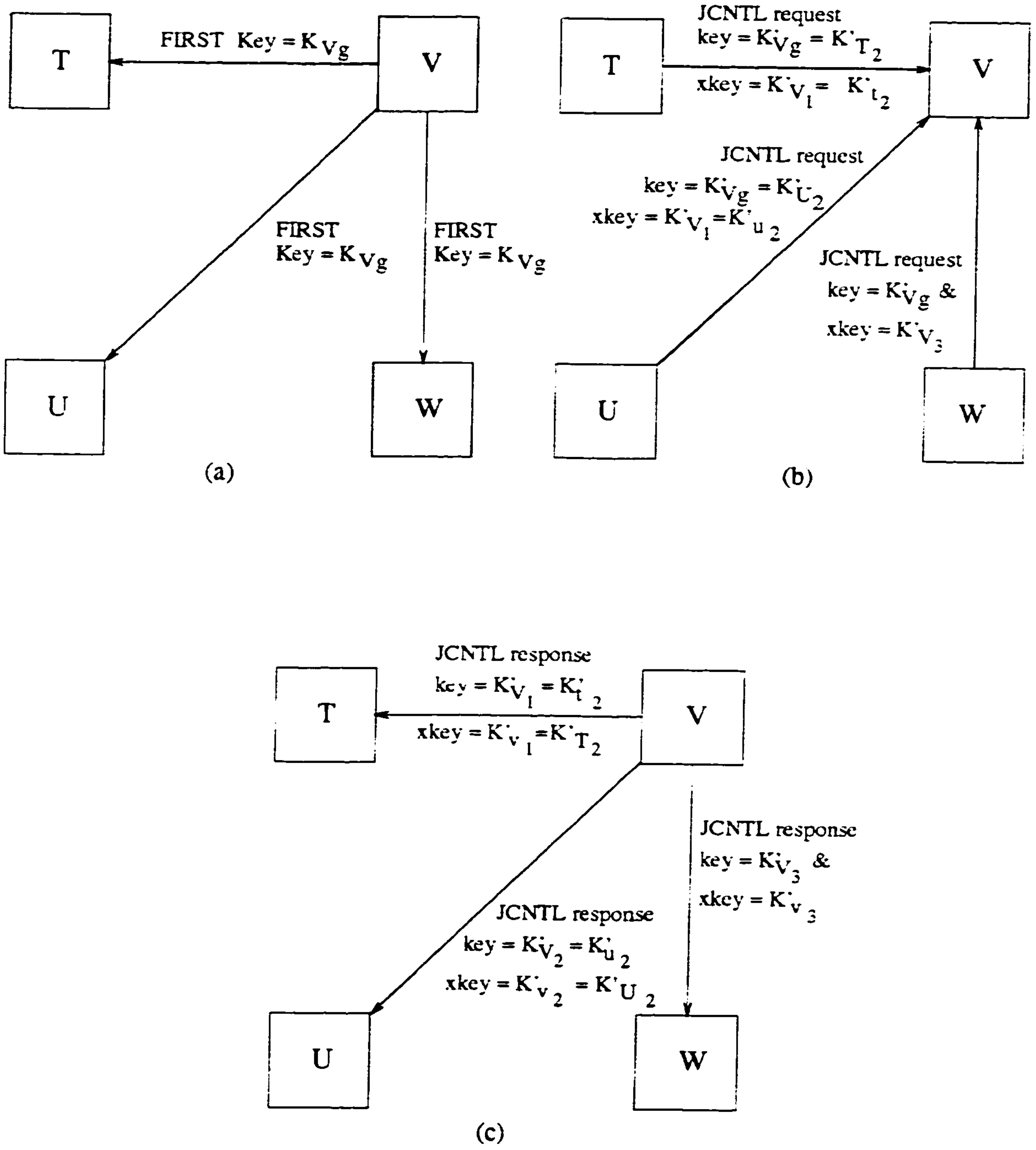

T
V
U
W
FIRST Key = K_{Vg}
FIRST Key = K_{Vg}
FIRST Key = K_{Vg}
(a)
JCNTL request key = $K'_{Vg} = K'_{T_2}$
xkey = $K'_{V_1} = K'_{t_2}$
JCNTL request key = $K'_{Vg} = K'_{U_2}$
xkey = $K'_{V_1} = K'_{u_2}$
JCNTL request key = K'_{Vg} & xkey = K'_{V_3}
(b)
JCNTL response key = $K'_{V_1} = K'_{t_2}$
xkey = $K'_{v_1} = K'_{T_2}$
JCNTL response key = K'_{V_3} & xkey = K'_{v_3}
JCNTL response key = $K'_{V_2} = K'_{u_2}$
xkey = $K'_{v_2} = K'_{U_2}$
(c)

Figure 24: Keys exchanged by host V when it wishes to transmit

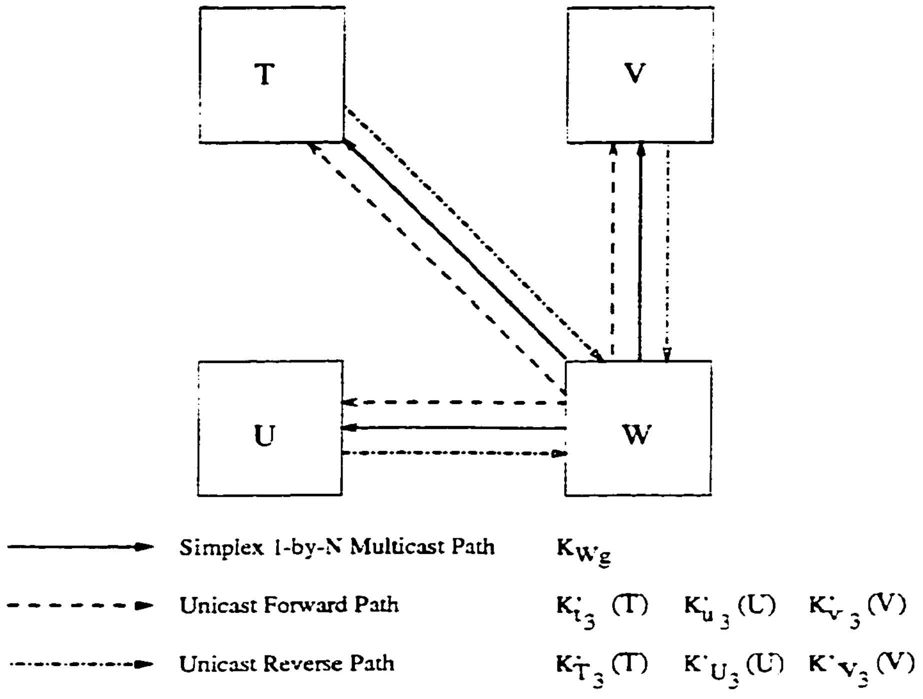
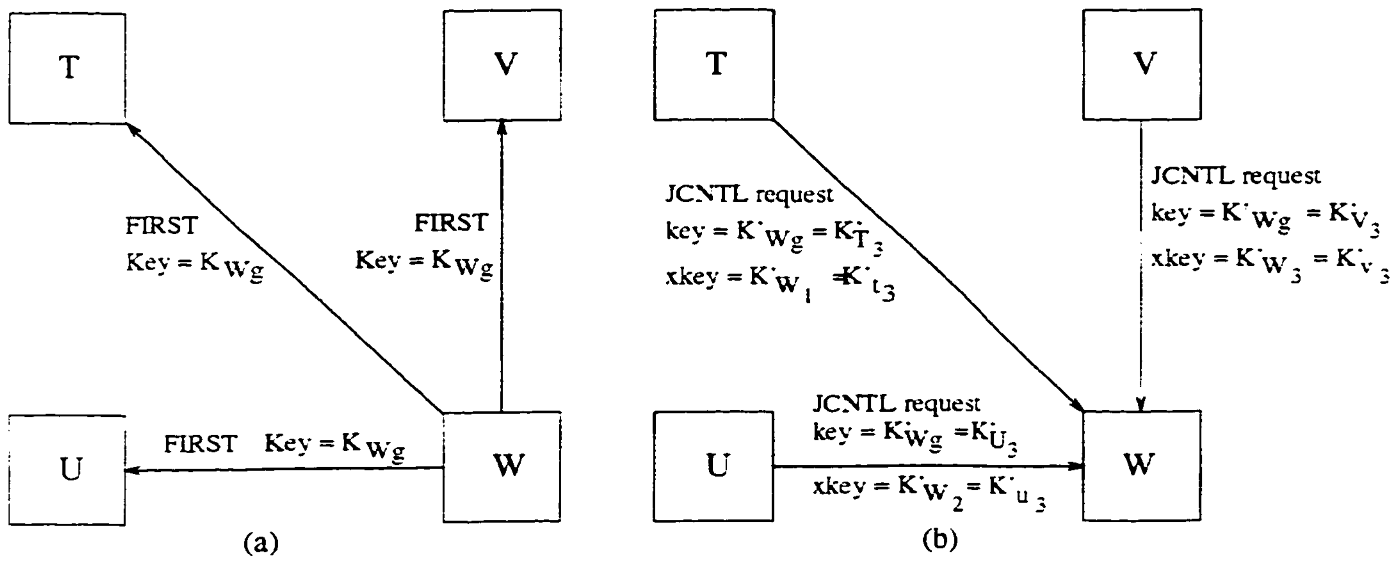

T
V
U
W
Simplex 1-by-N Multicast Path
K_{W_g}
Unicast Forward Path
K'_{T_3} (T) K'_{U_3} (U) K'_{V_3} (V)
Unicast Reverse Path
K'_{T_3} (T) K'_{U_3} (U) K'_{V_3} (V)

Figure 25: Data-Flow at the Transport Layer when process at V transmits

Transmitting Host	Group Key
T	K_{T_g}
U	K_{U_g}
V	K_{V_g}
W	K_{W_g}

Table 6: Key used by host when multicasting data to the M-by-N group



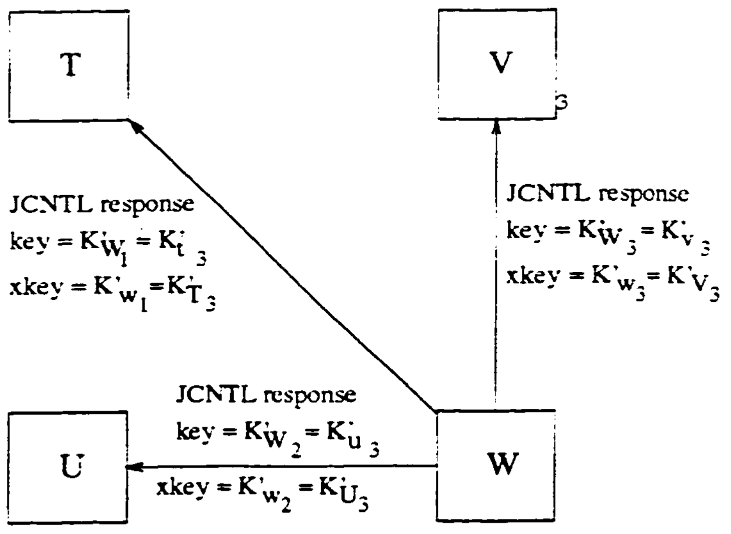


Figure 26: Keys exchanged by host W when it wishes to transmit

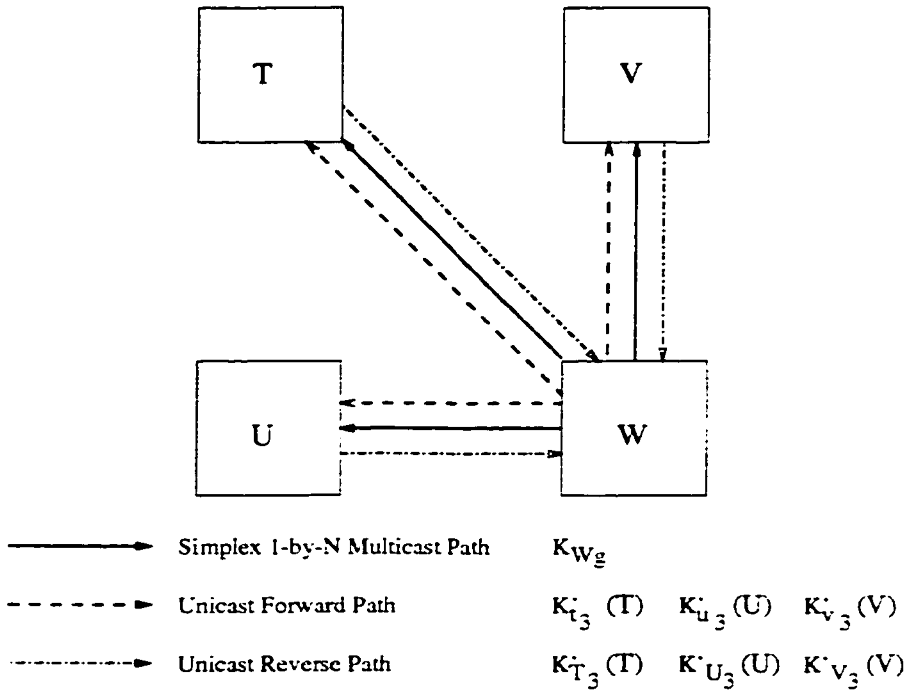

T
V
U
W
Simplex 1-by-N Multicast Path
K_{W_g}
Unicast Forward Path
K'_{t_3} (T) K'_{u_3} (U) K'_{v_3} (V)
Unicast Reverse Path
K'_{T_3} (T) K'_{U_3} (U) K'_{V_3} (V)

Figure 27: Data-Flow at the Transport Layer when process at W transmits

Transmitting Host	Receiving Host	Key
T	U	K_{T_1}
T	V	K_{T_2}
T	W	K_{T_3}
U	T	K_{t_1}
U	V	K_{U_2}
U	W	K_{U_3}
V	T	K_{t_2}
V	U	K_{u_2}
V	W	K_{V_3}
W	T	K_{t_3}
W	U	K_{u_3}
W	V	K_{v_3}

Table 7: Key used by host when unicasting data to other hosts

by the group members arrive in the same order at all recipients, even if they originate at different senders. The example above described how XTP's transport mechanism must be modified to accommodate true M-by-N communication. The ECSL, which sits on top of the XTP transport, must have mechanisms necessary for the necessary message exchanges. The next section looks at the design of such a support layer. It must be pointed out that the ECSL has been modelled on the lines of another well-known multipoint-to-multipoint transport protocol, MTP.

4.4 The Enhanced Communications Support Layer

For the design of the ECSL on top of XTP's modified M-by-N communication facility, we continue with our example from the previous section. The ECSL will incorporate a number of building blocks listed in the taxonomy and it will allow XTP to be efficiently used for the purpose of M-by-N communication.

As described in the previous chapter, the level of reliability required by the application determines the behaviour of our model. For instance, in case a host crashes or is partitioned away, then fully reliable multicast would require that all the data that was missed by that particular host be retransmitted in case the partitioned host rejoins the group. This would imply buffering of massive amounts of data by the ECSL. It would be more efficient to have the application to deal with such issues. Hence our present model does not provide the necessary mechanisms for dealing with partitions and late join of group members.

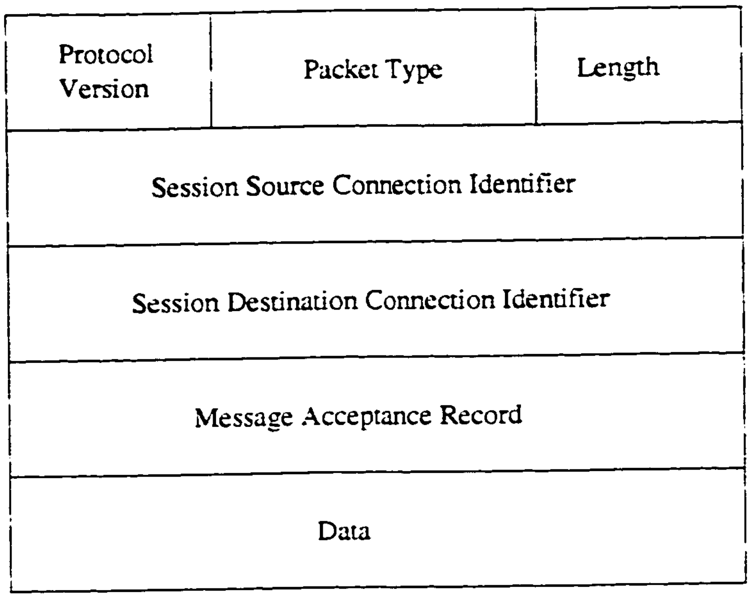
The ECSL provides a set of primitives that the application can use to participate in a multi-party communication. For instance, when a process wishes to transmit data to the group, it issues an open primitive to in turn call an open primitive at the transport interface. This results in a XTP FIRST packet (with the RCLOSE bit cleared) being sent to the other participating members. Then there is a series of

JCNTL packet exchanges resulting in a master context (association master context) and a set of subordinate contexts at T. Now a simplex 1-by-N multicast path exists for process A from host T to hosts U, V and W, with a full-duplex unicast path between host T and hosts U, V and W respectively. The previous section detailed the keys that are exchanged for the purpose of connection setup and data transfer at the transport layer.

Now if process B on Host U wishes to transmit data into the group, it first needs to acquire a transmit token from the master of the group using an ECSL token request primitive. This primitive in turn calls a transport token request primitive, which results in the transmission of a unicast Token request packet to the master. Depending on the implementation, the association master's ECSL may choose to pass the request up to the application in order to allow the user to accept or reject the member wishing to transmit, which in turn results in a Token response or Token deny packet. A more simplistic approach may be to just respond immediately with a token response packet, which is sent on the unicast path back to the requesting host. The token response packet will contain a list of members who are part of the active group as well as the ordering rule that needs to be followed.

Now process B calls an open connection primitive at the ECSL, which in turn results in a open transport connection primitive. After the required JCNTL packet exchanges are completed between U and T, V and W, there exists one master context and a set of slave contexts at host U. Hence a simplex 1-by-N multicast path is created between U and the other three nodes and bidirectional unicast paths are created from U to all the other nodes. The host master context handles all the 1-by-N multicast data flow in the forward direction while the subordinate contexts are responsible for reverse control packet flow (for the 1-by-N outgoing data), and the (reverse) data flow from the receiving entities.

From the above description, it is obvious that the ECSL service interface must


Protocol Version
Packet Type
Length
Session Source Connection Identifier
Session Destination Connection Identifier
Message Acceptance Record
Data

00 - Token Request Packet
01 - Token Response Packet
02 - Data Packet

Figure 28: Format of ECSL PDU

provide a number of packets to perform a minimum number of functions, such as starting up an M-by-N multicast session, distributing and managing the tokens that are required for transmission to the group and performing data transfer functions.

Figure 28 shows the format of an ECSL PDU. It consists of a ECSL protocol header followed by a variable amount of data. The header is part of every PDU.

The first 8 bits of the packet are the protocol version number. The discussion here refers to version 1 of the ECSL protocol and thus the field has the value of 0x01. The next twelve bits form the Packet Type. Table 8 gives a list of the different packets used by the ECSL protocol. The next segment is a 12 bit length of data field.

The ECSL source connection identifier is a 32-bit field containing the value assigned by the transmitting system when the ECSL was created. Similarly the ECSL destination connection identifier is a 32-bit identifier for the destination entity.

The next field is a 16 element vector, which represents the status of the last

Service	Primitive	Parameters
Open	S_OPEN_MASTER	Group membership and Ordering Rule
-	S_OPEN_SLAVE	Group membership and Ordering Rule
Token Request	S_TOK_REQ	-
Token Response	S_TOK_ACC	Active group list and Ordering Rule
Token Deny	S_TOK_DEN	-
Token Cancel	S_TOK_CAN	-
Data	S_DATA	-

Table 8: Enhanced Communications Support Layer Service Primitives

16 messages transmitted to the group. This field along with a 16 bit ECSL Token number field and ECSL packet number field make up a Message Acceptance Record (see figure 29). The message sequence numbers are 16-bit unsigned values. This field is initialised to zero by the master when the ECSL is initialised, and incremented by one after each token is granted. Only the master is permitted to change the value of the message sequence number. Once granted, that message sequence number is consumed and the state of the message must eventually become either accepted or rejected. Packet sequence numbers are unsigned 16 bit numbers assigned by the sending process on a per-message basis. Packet sequence numbers start at a value of zero for each new message and are incremented by one (consumed) for each data packet making up the message. The ECSL packet sequence numbers have a one-to-one mapping with the transport layer packet sequence numbers.

We define a message to be a concatenation of user data portions of a series of data packets with the last packet in the series carrying an end of message indication. A message may contain any number of bytes of user data, including zero. The ECSL Protocol must ensure that all processes agree on which messages are accepted and in what order they are accepted. The master controls this by allocating transmit tokens and setting the status of messages using a message acceptance record, which is carried in every packet. Other peer ECSL's then learn of the status of earlier messages by processing this information.

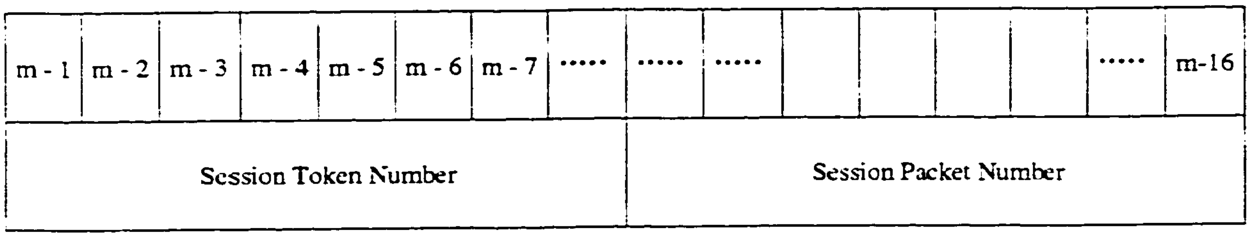


Figure 29: Format of a Message Acceptance Record (when token for message m is transmitted)

The first 32 bits of the record show the status of the 16 most recent messages, which the master sets according to the following rules.

- If the master has seen the entire message (including the EOM and all intervening data packets), the status is accepted.
- If the master has not seen the entire message but believes the message sender is still operational and connected to the master (as determined by the master), the status is pending.
- If the master has not seen the entire message but believes the sender to have failed or partitioned away, the status is marked as rejected.

The record also contains a 16-bit ECSL Token Number and a 16-bit ECSL Packet Number. The token number is incremented by one after each token is granted by the master. The packet number, which has a one-to-one mapping with the transport sequence number, is assigned by the producing process and is incremented by one on a per-packet basis.

The rest of the packet makes up the data portion of the packet. The last data packet needs to carry an end-of-message marker and is also used for implicitly surrendering the transmit token back to the master.

Chapter 5

Conclusion

The first goal of this thesis was to clearly distinguish the requirements from the mechanisms used to satisfy those requirements in the case of reliable multicast protocols. Having cleared that big quagmire, a classification was done to create a taxonomy of the various features. A few well-known multicast protocols were then chosen to be mapped over our taxonomy to demonstrate its completeness. The second goal of the thesis was to propose certain modifications to improve XTP's existing multipoint-to-multipoint communication model. Though our present design requires the use of at least M*N contexts, it only requires one multicast address. This is significantly better than most of the techniques that exist at present for achieving M-by-N multicast in XTP. Thus, we believe that the efforts in this thesis have gone towards successfully achieving the two listed goals.

5.1 Observations and Review of Work

It has become clear that the reliability that an application demands from a protocol determines many aspects of a protocol's design and subsequent performance, including its complexity, efficiency, overhead, and the amount of state that it requires senders and receivers to maintain. The type or level of reliability guaranteed by a protocol also determines how well suited it may be for certain types of applications.

It is claimed that the proposed design satisfies at least a subset of the requirements found in the taxonomy. To justify this claim, we revisit the ECSL design and map its features over one of the multicast transport protocols described in Chapter 3. For ease of explanation, we have chosen the Multicast Transport Protocol (MTP). As mentioned earlier, MTP provides sequencing and global ordering using the services of a central entity known as the coordinator. In our model for instance, the association master context plays the role of the coordinator. When one of the entities participating in the M-by-N multicast session wishes to transmit data, it must acquire a token from the coordinator using the token request primitive. The order in which the tokens are issued will determine the transmission order.

5.2 Current Limitations and Future Work

We believe that the claim of completeness in the taxonomy will be short-lived. As more powerful multicast applications emerge, the requirements they pose on the underlying protocols are sure to change. Advances in network technology are also likely to result in newer and better mechanisms to satisfy the necessary requirements. This will warrant updating the taxonomy to reflect these changes.

Another possible classification of the set of multicast protocols can be on the basis of their communication model. For instance, several protocols make use of hierarchy whilst others do not. Such a view has not been explored in this thesis due to lack of time.

As mentioned earlier, the needs of present day M-by-N type applications do not require scalability. However, this may be a requirement in the very near future. Hence addition of some sort of heirarchial mechanisms to improve scalability would ensure that our present model can provide a complete range of mechanisms to satisfy all the requirements mentioned in the taxonomy.

Another aspect that has not been looked into is the issue of late receiver join. One reason for this is the inability of most present day transport layers to buffer all the data since the beginning of a particular session so that a late joiner may catch up. This issue may be tackled with the help of some sort of checkpointing mechanism.

Bibliography

[Armi97] Grenville J. Armitage, *IP Multicasting over ATM Networks*, IEEE Journal on Selected Areas in Communication, Vol. 15, No. 3, April 1997, pp. 445 - 457

[Arms92] S. Armstrong, A. Freirer, K. Marzullo, *Multicast Transport Protocol*. RFC1301, February 1992

[Atwo96a] J. William Atwood, Octavian Catrina, John Fenton, and W. Timothy Strayer, *Reliable Multicasting in the Xpress Transport Protocol*. Proceedings of the 21st Local Computer Networks Conference (LCN '96), Minneapolis, MN, October 1996

[Atwo96b] J. William Atwood, *Large Group Reliable Multicasting in the Xpress Transport Protocol*. Proceedings of the 3rd International Workshop on Protocols for Multimedia Systems (PROMS '96), Madrid, Spain, October 1996

[Ball97] T. Ballardie, *Core Based Trees (CBT) Multicast Routing Architecture*, RFC2201, September 1997

[Borm94] Carsten Bormann, Jorg Ott, C.H. Gehrcke, T. Kerschat, Nils Seifert, *MTP-2: Towards Achieving the S.E.R.O. Properties for Multicast Transport*, International Conference on Computer Communications and Networks (ICCCN '94), September 1994

[Borm97] Carsten Bormann, Jorg Ott, Nils Seifert, *MTP/SO: Self-Organizing Multicast,* Internet Draft, Work in progress, November 1997

[Cain97] Brad Cain, Stephen E. Deering, Ajit Thyagarajan, *Internet Group Management Protocol, Version 3.* Internet Draft, Work in progress, November 1997

[Deer89] Stephen E. Deering, *Host Extensions for IP Multicasting.* RFC1112, August 1989

[Deer90] Stephen E. Deering, David R. Cheriton, *Multicast Routing in Datagram Internetworks and Extended LANs,* ACM Transactions on Computer Systems, Vol. 8, No. 2, May 1990, pp. 85 - 110

[Deer95] Stephen E. Deering, R. Hinden, *Internet Protocol, Version 6 (IPv6) Specification.* RFC1883, December 1995

[Deer96] Stephen E. Deering, Deborah L. Estrin, Dino Farinacci, Van Jacobson, Ching-Gung Liu, Liming Wei, *The PIM Architecture for Wide-Area Multicast Routing.* IEEE/ACM Transactions on Networking, Vol. 4, No. 2, April 1996, pp. 153 - 161

[Deer97] Stephen E. Deering, Deborah L. Estrin, Dino Farinacci, Van Jacobson, Ahmed Helmy, Liming Wei, *Protocol Independent Multicast Version 2, Dense Mode Specification.* Internet Draft, Work in progress, June 1998

[Diot97] Christophe Diot, Walid Dabbous and Jon Crowcroft, *Multipoint Communication: A Survey of Protocols, Functions and Mechanisms,* IEEE Journal on Selected Areas in Communications, Vol. 15, No. 3, April 1997, pp. 277 - 290

[Erik94] Hans Eriksson, *MBONE: The Multicast Backbone,* Communications of the ACM, Vol. 37, No. 8, August 1994, pp. 54 - 60

[Fenn97] W. Fenner, *Internet Group Management Protocol, Version 2,* RFC2236, November 1997

[Gane97] Ganesh Ramasivan, *Multicast: Rationale, Technology and Perspectives,* Technical Report available at http://www.cs.concordia.ca /~grad/ramasiv/mfinal.ps, August 1997

[Garc91] Hector Garcia-Molina, and Annemarie Spauster, *Ordered and Reliable Multicast Communication,* ACM Transactions on Computer Systems, Vol. 9, No. 3, August 1991, pp. 242 - 272

[Hand99] M. Handley, B. Whetten, R. Kermode, S. Floyd, L. Vicisano, *The Reliable Multicast Design Space for Bulk-Data Transfer.* Internet Draft, Work in progress, June 1999

[Hanna] Aiman Latif Hanna, *Masters Thesis, High Speed Protocols Laboratory, Department of Computer Science, Concordia University. Work in progress,* October 1999

[Hofm96] Markus Hofmann, *Adding Scalability to Transport Level Multicast,* Proceedings of Third International COST 237 Workshop - Multimedia Telecommunications and Applications, Barcelona, Spain, November 1996

[Hofm97] Markus Hofmann, *Enabling Group Communication in Global Networks,* Proceedings of Global Networking '97, Calgary, Alberta, Canada, Volume II, June 1997, pp. 321 - 330

[Jaco97] Van Jacobson, Mark Handley, Allyn Romanow, Lixia Zhang, Jean Bolot, *Requirements for Congestion Control for Reliable Multicast*, Slides from ftp://ftp.ee.lbl.gov/talks/sf-RMreq.ps, September 1997

[Mank98] A. Mankin, A. Romanow, S. Bradner, V. Paxson, *IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols*. RFC2357, June 1998

[Moy94] J. Moy, *Multicast Extensions to OSPF*, RFC1584, March 1994

[Obra98] Katia Obraczka, *Multicast Transport Protocols: A Survey and Taxonomy*, IEEE Communications Magazine, Jan. 1998, pp. 94 - 102

[Paul97] Sanjoy Paul, Krishnan K. Sabnani, John C. Lin, Supratik Bhattacharyya, *Reliable Multicast Transport Protocol (RMTP)*, IEEE Journal on Selected Areas in Communication, Vol. 15, No. 3, April 1997, pp. 407 - 421

[Pusa98] T. Pusateri, *Distance Vector Multicast Routing Protocol Version 3*, Internet Draft draft, Work in progress, April 1998

[QoS] Quality of Service Forum, *Papers and Related Links available at:* http://www.qosforum.com/tech_resources.htm

[Raja92] Bala Rajagopalan, *Reliability and Scaling Issues in Multicast Communications*, ACM SIGCOMM Computer Communication Review, Vol. 22, No. 4, October 1992, pp. 188 - 198

[SDR] Session Directory, *Papers and Related Links available at:* http://www-mice.cs.ucl.ac.uk/multimedia/software/sdr/

[SMuG] The Secure Multicast Research Group - hosted by Stardust Forums, *Papers and related links available at:* http://www.ipmulticast.com/ community/smug/

[Stra92] W. Timothy Strayer, Bert J. Dempsey, Alfred C. Weaver, *XTP: The Xpress Transfer Protocol,* Available from Addison-Wesley, ISBN 0-201-56351-7, 1992

[Wang97] Zheng Wang, John Crowcroft, Christophe Diot, Atanu Ghosh, *Framework for Reliable Multicast Application Design,* High Performance Protocol Architectures Workshop, 1997

[Whet99] B. Whetten, L. Vicisano, R. Kermode, S. Floyd, M. Handley, *The Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer,* Internet Draft, Work in progress, June 1999

[XTP3.6] Greg Chesson et al., *Xpress Transfer Protocol Definition Revision 3.6.* Protocol Engines Inc., SantaBarbara, CA, January 1992

[XTP4.0] W. Timothy Strayer et al., *Xpress Transfer Protocol Definition Revision 4.0.* XTP Forum, Santa Barbara, CA, March 1995

[XTP4.0b] W. Timothy Strayer. et al., *Xpress Transport Protocol Specification Revision 4.0b,* XTP Forum, Santa Barbara, CA, June 1998

[X6Rec93] Recommendation X.6, *Multicast Service Definition.* CCITT Recommendation, March 1993

[JTC1/SC6] ISO/IEC, *Information Technology - Enhanced Communications Transport Service Definition,* ISO/IEC FDIS 13252: 1999(E)

Appendix A

Multicasting in XTP

A.1 Introduction

There exists a plethora of applications (e.g., distributed databases, distributed simulations, multimedia teleconferencing and video-conferencing, sensor data distribution etc.,) that utlize the multicast communication paradigm, making multicast the most distinguishing and important feature of XTP.

XTP multicast provides a powerful set of mechanisms for group communication amongst a single sender and one or more receivers. Since this is transport layer multicast, flow, rate and error control procedures are applied to the transmission of arbitrary-size messages to arbitrary-size groups. XTP multicast provides the same control algorithms and mechanisms as XTP unicast but the basic difference between the two is the fact that while XTP unicast allows full-duplex data flow between the two communicating entities, XTP multicast is simplex data flow (sender to one-or-more receivers).

XTP multicast packets obey the same syntax rules as non-multicast packets and the packet structure is identical except that all packets in a multicast association have the MULTI bit set in the packet header. A multicast transmitter may utilize either the multicast group address or an individual transmitter's unicast address to transmit packets. A multicast receiver on the other hand must use the transmitter's unicast address to send control packets.

To indicate that a multicast association is inherently simplex (i.e., sender's incoming data stream is closed), the RCLOSE bit in the header is set for each outgoing packet from the sender, starting with the FIRST packet. All packets sent by the multicast receivers will have the WCLOSE bit in the header set.

The protocol assumes that the underlying data delivery service provides a multicast or broadcast service. It does not define how to assign and manage group addresses for this service and these details must be handled by mechanisms external to the protocol.

XTP provides a wide range of data reliability mechanisms from the UDP style best-effort multicast delivery to ordered and reliable multicast delivery. For reliable multicast transmission, an XTP transmitter must maintain and constantly update the state information of all the receivers in a group. In general, the transmitter needs to maintain state only for a subset of the receivers also known as "active" receivers, and run its control algorithms on them. The group management policy used by the transmitter determines the set of active receivers. Hence the active receivers are the ones whose control information is used to drive the control algorithms, while control information from all other receivers is not used. There exist orthogonal mechanisms for reliable management of information about group members.

The following gives a simplified description of the XTP multicast procedures as laid out in the latest version of the XTP specification [XTP4.0b]. The reader is asked to refer to this document for specific information such as packet formats and other relevant details about the protocol.

A.2 Multicast Group Creation and Termination

There are two different ways by which potential receivers may become part of a multicast association.

1. **Transmitter-Initiated Join:** This is an "invitation" to join the multicast association, where the transmitter initiates the join procedure by multicasting a FIRST packet to the group soliciting receivers. Any receiver listening on this address then responds with a JCNTL packet back to the transmitter, requesting to join the multicast association. If the transmitter wishes to allow that receiver to join, it responds with another JCNTL packet, which indicates that the receiver is part of the association.

2. **Receiver-Initiated Join:** This is also known as "late-join", in which case receivers try to take part in an ongoing association by transmitting a JCNTL packet to the group address requesting admission from the transmitter. The transmitter will then respond with another JCNTL packet allowing the receiver to join the association.

It is obvious that the transmitted-initiated join must always be used for any new session. Unlike unicast communication, multicast does not require the presence of receivers for commencement of a session. Hence it might be possible that an invitation to join does not yield any receivers and in such cases, the late-join procedure may be used.

If a multicast receiver wishes to withdraw from a group, it can do so by transmitting a CNTL packet with the END bit set. The multicast transmitter removes the receiver from the group but does not have to close the association even if that was the last receiver in the group. A transmitter can terminate a multicast association by sending a packet with the END bit set. Similar to the unicast case, various degrees of gracefulness are possible for closure of a multicast group.

A.3 Multicast Group Management

Multicast receivers are allowed to join or leave an ongoing session at any time, and hence multicast group management must ensure reliable multicast transmission in the face of such changes in membership of the multicast group. The reliability of a multicast association depends on the control algorithms used in the association and on the group management policy.

The XTP multicast transmitter maintains the state of the active multicast receivers as derived from control packets. The control algorithms running at the multicast transmitter make use of this information, while the group management policy determines when and how changes in membership are handled.

XTP does not impose policies for managing the receiver set because they are application and interface specific, but it provides the mechanisms for admitting, rejecting, and ejecting group members according to the group management policy. The specification defines three aspects to group communication where a policy is required.

1. **Group Membership Admission** The group membership policy should determine how the multicast transmitter will find its initial set of receivers. The policy should also determine who the active receivers are and are not.

2. **Group Membership Pruning** The policy should also dictate when a receiver should be removed from the active receiver group and when a receiver should be dropped from the receiver set altogether.

3. **Group Reliability** As the reliability of a multicast transmission depends on the group membership and in particular the responses from the active receiver group, the policy should determine if group reliability is compromised when the active receiver group becomes insufficient.

It is the responsibility of the multicast application to specify the parameters necessary to control the behaviour of the multicast association. The parameters include how the initial group of active receivers is compiled, the admission policy for the active receiver and regular receiver group during an in-progress association, the pruning or removal of receivers from the set of active receivers or from the group entirely and what is meant by receivers lagging behind other receivers.

The state information about each active receiver is maintained in multicast data structures at the multicast transmitter. This information can be regularly updated by the transmitter, by setting the SREQ bit in any outgoing packet, which forces receivers to respond with CNTL packets.

The transmitter can ensure syncronization amongst all receivers with respect to the data stream by observing the "rseq" values in the data structure to verify that all values are within some user-defined threshold of the transmitter's last "seq" value. A significantly low rseq value for a particular receiver might indicate that receiver is slower than the other members of the group, or just that that receiver has left the group without notification or died. To verify the status of such a receiver, the transmitter forces an update of state information using SREQ. If all receivers respond, then the offending receiver is just slow. However, if one or more receivers fail to respond after the expiry of WTIMER, a synchronizing handshake procedure will confirm whether the receiver is alive or dead.

The level of reliability required by an application determines the integrity of a group. The amount and type of information recorded in the group membership data structures, and the degree to which that information is exposed to the user, affects the breadth of group reliability semantics that can be imposed by the user. For an in-depth discussion about the requirements for group reliability, see section 3.1.1.1.2 on page 29.

A.4 Multicast Association Management

Managing an XTP multicast association is closely related to managing a group of receivers. The rest of the discussion details how a multicast association is established, maintained and terminated and what specific packets are exchanged for these actions.

A.4.1 Multicast Association Establishment

Similar to the unicast association establishment, a multicast association is established when one or more listening contexts receive a first packet and all the participating contexts (one transmitting and one or more receiving) move into the active state. As there may be multiple multicast contexts on the same host listening on the same multicast address, an incoming packet has to be matched against all the listening contexts to find those that will accept the association.

Each listening context submits an address filter that represents the address values that the context is willing to accept, as well as acceptable traffic shaping parameters and option bits. Once a FIRST packet is received and is not a duplicate for an already active context, it is subjected to a matching algorithm to determine if any listening contexts should get a copy of that packet. The contents of the FIRST packet are compared against each listening context's criterion for acceptance until all listening contexts have been examined. A copy of the FIRST packet is given to each listening context for which the address, traffic specification, and options are acceptable. Each listening context that accepts the FIRST packet also makes an entry for that context in its translation map. The entry will help the context map any further incoming packets whose "key" field is the same as this FIRST packet's key field, and whose source host's address (which can be obtained from the underlying data delivery service) is the same as this FIRST packet's source host's address. Now all the participating contexts move into an active state.

Notation	Explanation
Kg	Transmitter's Local Key for the multicast group
Kr	Receiver's Local Key
Ki	Key assigned by the Transmitter to uniquely identify a receiver i
Kg'	Transmitter's Local Key for the multicast group as the Return Key
Kr'	Receiver's Local Key as the Return Key
Ki'	Key assigned by the Transmitter to uniquely identify a receiver i as the Return Key

Table 9: Explanation of Notations used for Local and Return Keys

If the FIRST packet finds an active context, then it is a duplicate which might have been resent because the transmitting context timed out too early (due to an incorrect WTIMER value). In such a case, a copy of the FIRST packet should be given to each of the contexts to which this packet belongs, and each of them should respond accordingly if the SREQ or DREQ bits are set, and accept any additional data that is part of the FIRST packet, but no new contexts become active.

If a full context lookup finds neither listening nor active contexts, then the packet is dropped. This is very different from the unicast case where a rejected FIRST packet results in a DIAG back to the sender. The reason for this is due to the possibility that a multicast session may be active, even though there may be no interested receivers in a group at a particular point in time.

A.4.2 Multicast Packet Exchanges

The following explains the necessary communication and packet exchanges for XTP multicast. The procedures for both unreliable and reliable group information are dealt with for both the transmitter-initiated and receiver-initiated join procedures. The diagrams are drawn where necessary to indicate the basic context information and other important fields in the relevant XTP packets. Tables 9 and 10 give a description of the notations used in the following figures.

Notation	Explanation
SA	Source Address Field in an Address Segment
DA	Destination Address Field in an Address Segment
src	Source Address used by the Delivery Service
dest	Destination Address used by the Delivery Service
TAg(m)	Multicast Transport Address of the Group
TAt(u)	Unicast Transport Address of the Transmitter
TAr(u)	Unicast Transport Address of the Receiver
At(u)	Unicast Delivery Service Address of the Transmitter
Ar(u)	Unicast Delivery Service Address of the Receiver

Table 10: Explanation of Notations used for Addresses

A.4.2.1 Transmitter-Initiated Multicast

In a transmitter-initiated join procedure, the transmitter sends out a FIRST packet to the group (Refer figure 30) containing the transmitter's local key, the multicast transport address of the group and the unicast transport address of the transmitter[1].

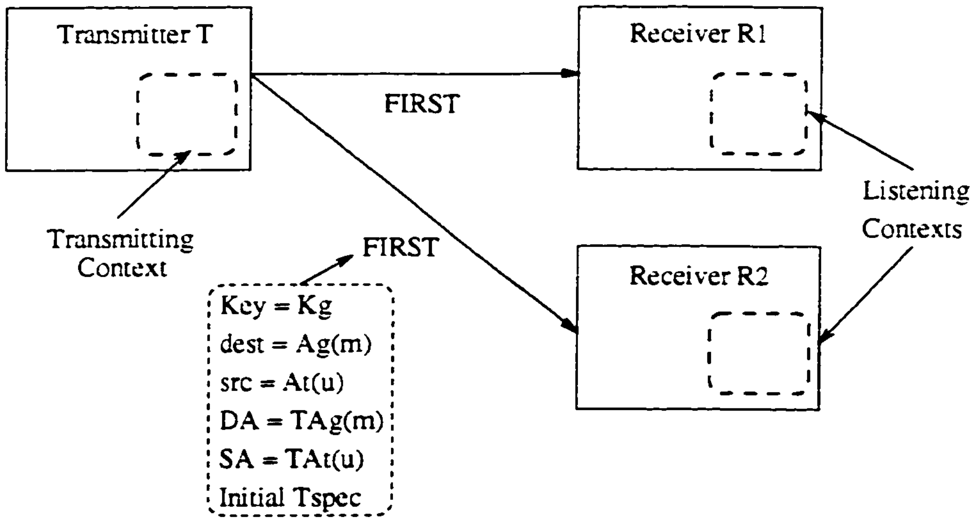


Figure 30: FIRST packet Soliciting Receivers to Join the Group

Once the FIRST packet arrives at a receiver, a entry is made in the receiver's

[1]The explanation refers to only certain fields of the FIRST packet and assumes that the other fields are present.

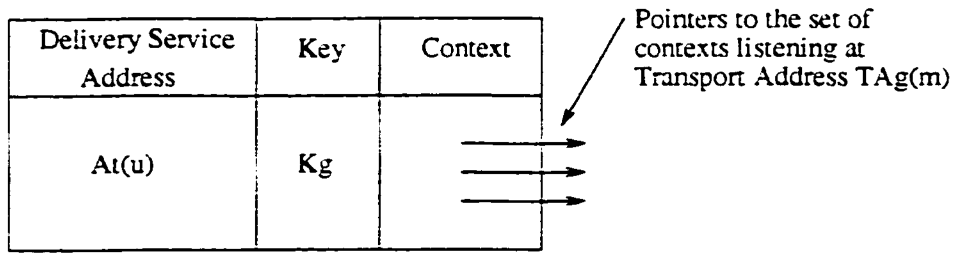

Delivery Service Address
Key
Context
At(u)
Kg
Pointers to the set of contexts listening at Transport Address TAg(m)

Figure 31: Translation Table Entry at each Receiver after receipt of FIRST packet

translation table to map future packets from that transmitter for that particular group address to the respective set of receiving contexts (Refer figure 31). It should be noted that though Figure 31 indicates pointers to listening contexts, it is possible to store the actual contexts there instead of pointers to them and such a choice is implementation-dependent.

A.4.2.1.1 Unreliable Groups

If the transmitter does not care about who its receivers are, it will not set the SREQ bit in the FIRST packet. Thus the receiver will remain silent. However this does not prevent the transmitter from setting the SREQ/DREQ bits in future outgoing packets to gather responses from listening receivers and using these responses to advance its outgoing sequence numbers. These responses must be returned with the transmitter's local key as the return key and the transmitter must make use of some algorithm to coalesce the responses. Hence in this case, though data reliability can be guaranteed, reliable reception by a defined group of receivers is not guaranteed.

A.4.2.1.2 Reliable Groups

If the transmitter wishes to have reliable knowledge of its receiver set, then the receivers need to send back control packets to the transmitter. In this case, the transmitter sets the SREQ bit in the outgoing FIRST packet inviting receivers to join the multicast association. Receivers interested in being part of this association

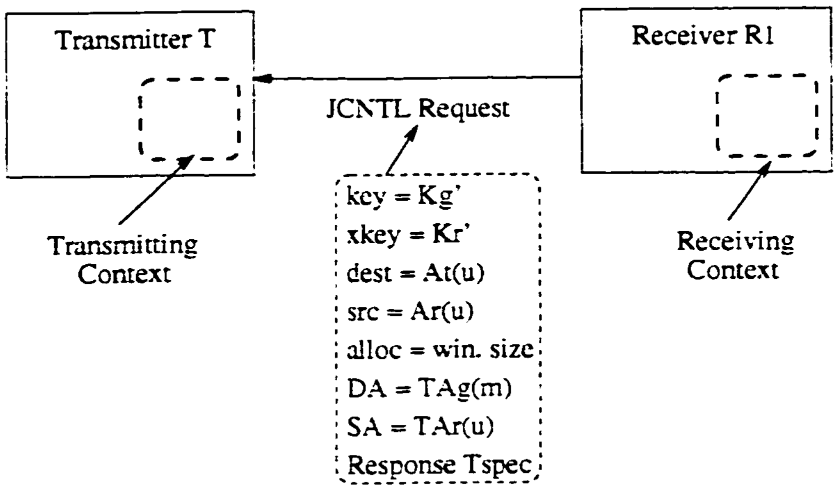


Figure 32: JCNTL Request Packet from Receiver

each responds with a JCNTL packet (Refer figure 32). This is a request to join the group and is directed to the unicast address of the transmitter containing the transmitter's local key (as return key) and the receiver's local key (as return key) for the purpose of key exchange. Though the request packet is a response to a SREQ in the first packet, for reliable group semantics, the SREQ is set in this packet and protected with the WTIMER. The alloc value which represents the amount of data a sender is permitted to transmit is set to the current window size of the receiver. At this point, the transmitter has the receiver's local key (as return key) and the receiver's transport address to uniquely identify each receiver in the group.

In order to complete the packet exchange, the transmitter issues a JCNTL response packet allowing the receivers who sent out JCNTL requests to join the association. This packet does not have the SREQ bit set. Depending on whether the transmitter would wish to continue to uniquely identify the receivers or not, one of two listed cases may arise.

Case I: In case the transmitter wishes to continue to identify its set of receivers, it assigns a key to uniquely identify each receiver and sends it out in the JCNTL response packet (Refer figure 33). When the JCNTL response packet arrives

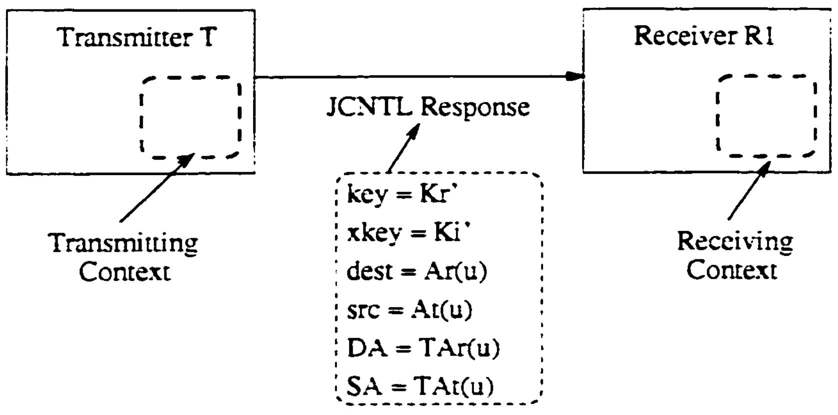


Figure 33: JCNTL Response Packet from Transmitter (Case I)

at the receiving host, it records that key as the key to be used in future for sending control packets back to the transmitter and adds an entry to its translation table to ensure correct delivery of a DIAG packet which is sent by the transmitter in case the transmitting context dies.

If the contents of the destination address field in the address segment of the JCNTL response packet does not match the unicast transport address associated with a particular receiver's local key, then a protocol error has occurred. This error is signalled by a DIAG packet to the transmitter with the receiver's local key. In order for the transmitter to know which receiver sent this DIAG, it adds a mapping to its translation table to identify that particular receiver (Refer figure 34).

Delivery Service Address	Key	Context
Ar(u)	Kr	Ki

Figure 34: Translation Table Entry at the Transmitter

It is possible that a transmitting context might cease to exist at any time. In such a case, instead of waiting for WTIMER to expire, the transmitter will send a DIAG packet with the individual receiver's key. To enable correct delivery of this

Delivery Service Address	Key	Context
At(u)	Ki	Kr

Figure 35: Translation Table Entry at each Receiver

DIAG to the appropriate context, the receiver makes an entry in its translation table to map that key assigned by the transmitter to uniquely identify a receiver with that particular receiving context (Refer figure 35).

Case II: In case the transmitter does not wish to continue to uniquely identify each receiver after receiving JCNTL requests packets from all those receivers, it sends a JCNTL response packet to all the receivers with the exchange key as the transmitter's local key (as return key) for the group (Refer figure 36).

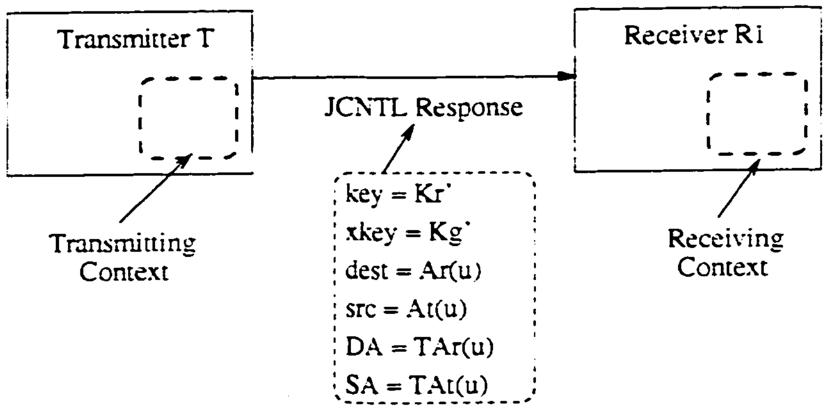


Figure 36: JCNTL Response Packet from Transmitter (Case II)

Once all relevant packet exchanges have been done and all necessary information recorded at both the transmitter and receivers, the transmitter may begin transmitting data packets to the group using its local key for the group. Such packets will always require a full-context lookup at the receiving host (See Figure 31).

If the transmitter wishes to send packets to an individual receiver it will use the receiver's local key (as return key) and the receiver needs to perform only an

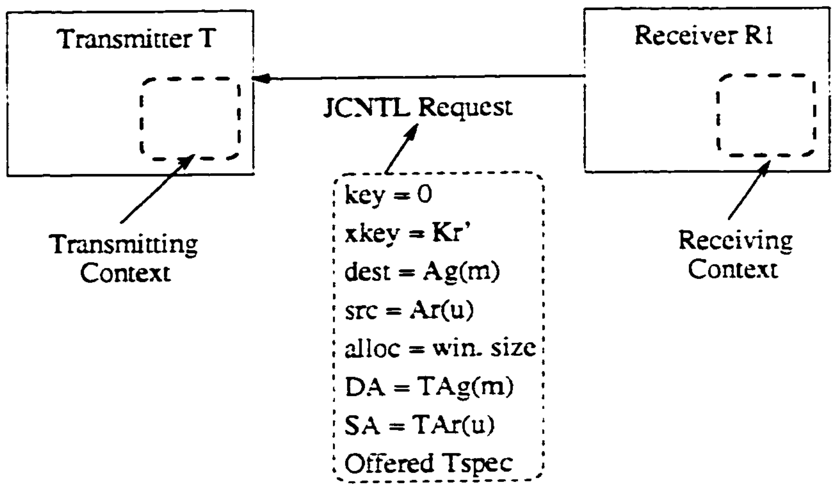


Figure 37: JCNTL Request Packet from Receiver

abbreviated context lookup to match these packets with their appropriate set of active contexts.

If the receiver wishes to send control packets back to the transmitter it may make use of the key that it learned from the initial FIRST packet (in which case the transmitter will not be able to identify the particular process at the host which sent the packet), or the key that it learned from the JCNTL response packet (in which case the transmitter will be able to precisely identify the process at the particular receiving host which sent the packet). The transmitter uses an abbreviated context lookup to map the appropriate entry.

A.4.2.2 Receiver-Initiated Multicast

When a receiver wishes to late-join an existing multicast association, it sends a JC-NTL request packet to the transmitter with the SREQ bit set (Refer figure 37).

The transmitter then responds with a JCNTL response packet with key values depending on whether the transmitter wishes to uniquely identify each new receiver or not.

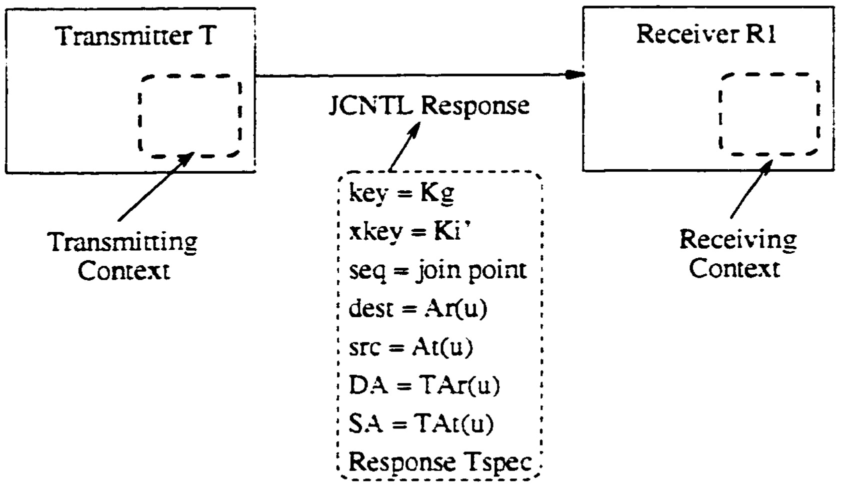


Figure 38: JCNTL Response Packet from Transmitter (Case I)

In case the transmitter wishes to uniquely identify each new receiver, it allocates an unique key to identify that receiver from its key space and sends this key as a return key in the exchange key field of the packet (Refer Figure 38). The transmitter also makes an entry in its translation table as before (Refer Figure 38).

If the transmitter does not wish to uniquely identify each new receiver, it uses the local key of the group (as return key) in the exchange key field of the packet (Refer Figure 39). The other procedures followed are similar to the transmitter-initiated join case.

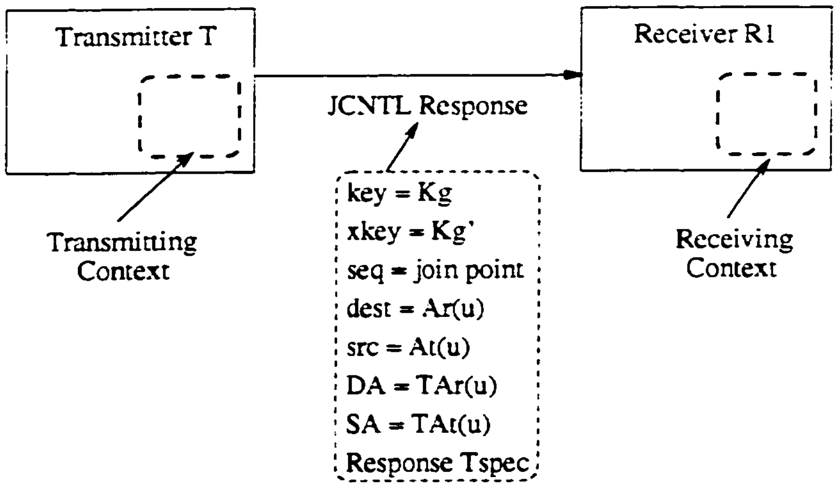


Figure 39: JCNTL Response Packet from Transmitter (Case II)

There are certain exceptional situations in the protocol in which case, packet exchanges are complicated by lost packets or simultaneous events. Readers are encouraged to refer to the specification for those details.

A.4.3 Multicast Association Termination

The transmitter may terminate associations or even entire sessions at any point in time. Individual receivers are also allowed to leave multicast associations at will.

When a receiver is no longer interested in the multicast association (because the application exited), the receiving context will voluntarily exit the association by sending a control packet to the transmitter with the END bit set, and move into a zombie state for a time period. In the zombie state, the multicast receiver responds only to packets sent to its unicast address with a return key in its key field. The response is typically a retransmission of the control packet with the END bit set. The multicast transmitter updates its active receiver group if any active receivers leave the association. The behaviour of the protocol also depends on the group reliability semantics.

A multicast receiver may also be forced to leave the association at any time. If a multicast receiver receives a packet with the END bit set, it must immediately abandon the association and move into a quiescent state.

If an active multicast receiver fails to respond to a status request from the transmitter, the transmitter will timeout and enter into a synchronizing handshake procedure. If the receiver fails to respond even after this time period, it is assumed to be dead and is removed from the list of active receivers.

A.5 Flow, Rate and Error Control Procedures

A multicast transmitter follows the same rules for flow control, rate control and error control as a unicast sender. It uses default or inherited values for flow, rate and error control algorithms and takes new values from receiver responses to SREQ and DREQ bits set in the header of outgoing DATA or control packets by the transmitter. The sender also periodically updates the round-trip time estimate by observation.

The only consideration specific to multicast is that the values for flow, rate and error control must be resolved in such as way that they are an aggregate of all values from the known group of receivers. The decision as to how these values are aggregated from the receiver group is implementation, and possibly application specific, and hence is not defined by the protocol.

As with unicast, multicast error control is based on the exchange of information regarding lost or damaged data and the retransmission of this data. Depending on whether the NOCHECK bit is set in the header of a packet, the checksum is performed over the header only or over the whole packet. Data loss and recovery is via a standard acknowledgement and retransmission procedure and timers are used to detect loss of status requests. Recovery in such a case is through an exchange of packets designed to resynchronize the endpoints of the association (also called a synchronizing handshake). Other error conditions including protocol errors are indicated using DIAG packets which are unicast to the respective hosts with a return key in the key field.