A FRAMEWORK FOR RELIABLE MULTICAST PROTOCOL

By

VENKATA LAKSHMANAN RAMASUBRAMANIAM

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2002

To

My Parents

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Dr. Richard E. Newman for giving me the opportunity to work with him and being my advisor. He has been a great source of inspiration and encouragement throughout my stay at the University of Florida.

I would also like to thank Dr. Randy Y. C. Chow, Dr. Jonathan C. Liu, and Dr. Michael P. Frank for serving on my committee.

Special thanks go to Dr. Christophe Fiorio for generously allowing me to use his LaTeX algorithm template. Thanks go to Mr. Ron Smith for creating this thesis template for the LaTeX community at University of Florida and helping me out in the various problems encountered during the formatting.

I would like to thank my parents, family and friends who have always been there to help me.

TABLE OF CONTENTS

LIST OF FIGURES

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

A FRAMEWORK FOR RELIABLE MULTICAST PROTOCOL

By

Venkata Lakshmanan Ramasubramaniam

December 2002

Chair: Richard E. Newman
Major Department: Computer and Information Science and Engineering

The biggest revolution in networking since the introduction of the World Wide Web (WWW) is Internet Protocol (IP) multicasting. Multicasting is becoming increasingly ubiquitous in today's Internet world. There are many commercial applications, both real-time and non real-time, that are based on IP Multicasting. Many of those applications require reliable delivery of data at the receiver's end for them to be meaningful.

Reliable multicast refers to the reliable manner in which a message should reach the group of receivers. It is still an active area of research and has a number of protocols in existence. There are several issues like scalability, error recovery and congestion control that make the design of a reliable multicast protocol difficult. This thesis discusses various reliable multicast protocols in existence and provides a framework for a scalable, reliable, dynamic multicast transport protocol for point to multipoint communication.

The proposed protocol provides a sequenced, loss less delivery of data from a sender to a set of receivers. The receivers are split as logical groups with a special receiver called Group Leader controlling the group. The error recovery

and retransmissions are performed by the Group Leader, which distributes the processing load of the sender. The logical grouping of the receivers makes the protocol scalable by its architectural design. The protocol provides congestion control algorithms and also mentions the group formation and management techniques. A unique feature of the protocol is the ability of the receivers to reconfigure them to suit the network conditions in case of congestion.

CHAPTER 1
INTRODUCTION

Multicasting provides an efficient mechanism for message delivery from a single sender to a group of receivers. Multicasting is done using Internet Protocol (IP) [1], which is an unreliable protocol. Therefore, applications using multicasting that require reliable delivery of data have to use a reliable transport protocol. This is similar to reliable unicast applications using Transmission Control Protocol (TCP) [2] as its transport protocol.

Unlike TCP, the design of a transport protocol for multicasting imposes several challenges. The key issues include the implosion problem, scalability, error recovery and congestion/flow control. The presence of multiple receivers causes all the control information from the receivers to flow to the sender, that causes the implosion problem. The operation of the protocol should not deteriorate with the increase in the number of receivers. Due to the large number of receivers, the recovery time should be minimized and should have facilities for congestion and flow control. These topics are discussed in detail in Chapter 3.

There are a number of protocols in existence today for reliable multicasting: Scalable Reliable Multicast [3], Xpress Transport Protocol [4], Log-based Receiver Reliable Multicast Protocol [5], Local Group based Multicast Protocol [6] and Multicast TCP [7]. Some of the protocols like Reliable Multicast Transport Protocol [8], developed by Lucent Technologies, and Tree-based Multicast Protocol [9], developed by Sun Microsystems, are becomming commercially available. Chapter 2 surveys these protocols in detail. Although many transport protocols exist for reliable multicasting, none of them are standardized so far. The protocols, that are currently available have been designed to suit the needs of a custom

application; hence there is an absence of a generalized protocol that can be used by all the applications fairly. Also most of the protocols do not address important issues such as congestion control, which plays a major role due to the traffic generated by reliable multicasting. Some of the protocols are not scalable to a very large number of receivers, which is required of any multicast application.

1.1 Motivation

The Internet has been emerging as one of the major source of communication utility. This has made a dramatic effect in the way in which people communicate and share information with each other. A form of group communication called multicasting has led to the development of a number of distributed applications like video conferencing, distributed interactive simulation, white boards, electronic publishing, news/news group services, digital libraries, etcetera. Due to the immense requirement for security in the commercial applications of the Internet world, there was a huge amount of research in cryptography and standards like RSA [10] have been put forth. Similarly due to the emerging above-mentioned group communication-based applications, there is again the rise in interest for reliability.

Most of the real-time distributed applications require reliable delivery of data. Unlike unicasting, in which the data is exchanged between a single source and a receiver, exchanging the data reliably in a multicasting environment with a number of receivers poses a lot of challenges. The protocol's architecture, design and operation will play a major role in the operation of the same. Reliable multicast is now an actively researched area in the Internet community. Although a number of reliable multicast protocols exist for research and commercial applications, none of them are standardized. The reliable multicast protocols in existence today are designed for custom applications. Also, most of the protocols do not address some key issues like congestion/flow control and some of them are not scalable to a large

number of receivers, which typify mainly multicast applications. Hence there is a need for a reliable multicast transport protocol, which addresses all of the issues like scalability, error recovery, congestion/flow control and group management techniques and must be suitable for a wide variety of applications.

1.2 Problem Definition

This thesis proposes a framework for a reliable multicast transport protocol. Reliable multicast refers to the reliable manner in which a sender sends the data to a set of receivers. The protocol proposed in this framework provides a sequenced, loss less delivery of data from a single sender to a set of receivers. The framework addresses all the major issues like error recovery, congestion control, scalability, and group management techniques.

The architectural design of the protocol makes it scalable to a large number of receivers and aids local error recovery. The logical grouping of the receivers in the protocol makes the error recovery distributed. The logical grouping of receivers aid in combining the receivers with similar characteristics. The receivers that reside in the same location, receivers that have identical data rate, and the receivers that have identical error rate could be combined together. By making the error recovery distributed, the communication load of the network is greatly reduced. Receivers do not have to send their repair request all the way up to the source of the multicast tree and similarly the sender do not have to send the response down the tree to receiver. The control traffic involved in the error recovery is limited only to the logical groups. As the receivers get repaired from the Group Leader rather than the sender, the end-to-end latency is greatly reduced. The distributed error recovery also reduces the processing load of the sender as it is distributed among the Group Leaders. The retransmission performed by the Group Leader is sometimes multicast to the whole group that slightly reduces the processing load of the Group Leader.

The protocol responds to the congestion state in the network through TCP-like congestion control algorithms. The protocol is also highly dynamic with the multicast group receivers adapting themselves to the changing network conditions and congestion state. Dynamic reconfiguration of the receivers among the logical groups does this. The details of the protocol's architecture, design and operation are discussed in Chapter 4.

1.3 Organization of the Thesis

Chapter 2 discusses the multicasting techniques and evaluates the various existing protocols. Chapter 3 discusses the various issues pertaining to the design of reliable multicasting protocols and analyses the existing reliable multicast protocols pertaining to these issues. Chapter 4 presents the framework for the protocol, discussing its architecture, design and operation. Chapter 5 presents the functional model of the protocol and Chapter 6 gives the concluding remarks and suggestions for future work.

CHAPTER 2
BACKGROUND AND PREVIOUS WORK

This chapter gives an introduction to reliable multicasting and the various prominent works and protocols developed on the same.

2.1 Multicasting

Multicast is an efficient way to transfer data from a source to a group of receivers. Instead of sending a separate copy to each of the receivers, the sender sends to the network, a single copy, which then sends it to all the receivers. Roca *et al.* have surveyed different multicast technologies [11]. Multicasting is broadly classified into the following types based on the applications [12].

- *One-to-Many (1toM)*: The 1toM multicast technique has a single host sending data to more than one receiver. The one-to-many applications include scheduled audio/video distribution (lectures, presentations, meetings), push media applications (news headlines, weather updates, sports scores), file distribution and caching, announcements (network time, session information, keys) and monitoring applications (stock prices, sensor equipment, security systems).

- *Many-to-Many (MtoM)*: The MtoM multicast technique supports the multiple receivers acting as senders too, enabling two-way communication. The many-to-many applications include multimedia conferencing (audio/video, white board), concurrent processing (distributed parallel processing), distance learning, chat groups, distributive interactive simulation and collaboration (shared document editing).

- *Many-to-One (Mto1)*: The Mto1 multicast technique has multiple senders sending data to a single receiver. The many-to-one applications include data collection applications (sensors), auctions and polling.

With many of the multicast applications listed above, the growth of IP Multicast has grown to a great extent in the past few years. Of the many applications listed above, most of them require reliable delivery of data at the receiver's end.

2.2 Reliable Multicasting

In multimedia applications, the loss of some data could be acceptable - the video frames could be sacrificed for the audio information. The main objective of these applications is to guarantee the quality of service at the cost of reliability. But most of the applications discussed above require reliable delivery of data at the receiver end. Addition of the word *reliable* to multicasting imposes several challenges in the way the protocol is designed. In unicast communication between a single source and a receiver, reliability is provided by TCP [2]. TCP provides reliable transmission with feedback from the receiver, the source accordingly provides the retransmission of the missed packets. In multicasting, due to the presence of a large number of receivers, closed loop feedback results in implosion problem as discussed in Chapter 3.

The term reliable multicasting refers to the reliable manner in which the message should reach the group of receivers. The protocol developed should also be scalable to a large number of receivers. Earlier multicast protocols were broadly classified as **sender-initiated** and **receiver-initiated** [13]. In the case of sender-initiated protocols, the sender will maintain considerable state information. All the receivers sending in their ACKs, will lead to packet implosion. The presence of large number of receivers causes them to report their control information and acknowledgements to the source, that results in virtually impounding it. This is called as the implosion problem. In the case of receiver-initiated protocols, every receiver will maintain state information thereby shifting the burden from the sender to the receivers. There has been much comparative analysis done on the sender-initiated and receiver-initiated reliable multicast protocols. These analysis have shown that receiver-initiated protocols are more scalable than the sender-initiated ones. The protocol framework discussed in this thesis is receiver-initiated with slight

modification. Instead of all the receivers maintaining the state information, only selected receivers called Group Leaders will maintain the same.

2.3 Existing Protocols

A considerable amount of work has been reported in the literature regarding reliable multicast protocols. Most of the works could be classified as the sender-initiated and the receiver-initiated approaches. Although there are many protocols in the reliable multicast area, this section deals only with the major ones, discussing the highlights of the same. Most of the protocols are based on the logical groupings of the receivers. A set of receivers is grouped together to form logical groups that form a hierarchical multicast tree. The architecture improves the scalability of the protocol. Some of the protocols discussed do not form any groupings and their scalability is limited. Obraczka provides a comparison chart of all the major multicast transport protocols [14].

2.3.1 Local Group Concept

Markus Hofmann proposed a tree-based approach using the so-called Local Group Concept (LGC) [6]. The concept, developed in 1994 has evolved into two separate protocols: Local Group based Multicast Protocol (LGMP) and Local Group Configuration Protocol (LGCP). LGC is one of the few earlier protocols that discussed the local logical groupings of receivers.

LGMP is also based on the principle of local subgrouping. It has special nodes called group controllers that are responsible for local re-transmissions and ACK processing. The selection of the group controller and the management of the local groups is not a task of the multicast protocol like LGMP. Instead, a separate configuration protocol called as Dynamic Configuration Protocol (DCP) was designed and implemented. Any error or retransmissions are recovered in the local groups by the group controller. If any of the group receivers do not have the data packet, it is requested from the source. Acknowledgement schemes include a

positive, negative and a novel semi-negative acknowledgement scheme, which indicates that a data unit has not yet been received correctly, but it does not request the data unit for retransmission. Error recovery is first performed locally within the groups. Any member of the local group that has the missing packet would retransmit it to the requesting member. A packet is requested for retransmission from the sender only when it is not found with any of the members in the group, including the group controller. LGC defines two different modes of performing local retransmissions: load-sensitive mode and delay-sensitive mode. In load-sensitive mode, retransmissions are performed to minimize the network load. The controller withholds the retransmission waiting for requests from other receivers. In delay-sensitive mode, the retransmissions are performed immediately after the reception of retransmission requests.

The establishment and maintenance of logically structured group hierarchies is left to the Dynamic Configuration Protocol. The major advantage of the DCP is that it can interact with any other protocol that requires a logically structured receiver hierarchy. The DCP and LGC together form the protocol architecture. LGC concerns group formation, group characteristics and dynamic reconfiguration. But there is no mention of any congestion control scheme. The LGMP operates on top of User Datagram Protocol (UDP) [15] and does not require any changes within internal network equipment such as routers or switches.

2.3.2 Reliable Multicast Transport Protocol (RMTP)

Paul and Sabnani of Bell Laboratories proposed the Reliable Multicast Transport Protocol (RMTP) [8]. RMTP is commercially marketed by Lucent Technologies. In RMTP, the receivers form a dynamic multicast tree with the source rooted on top of the tree. RMTP is a protocol for point-to-multipoint reliable multicast.

The receivers are grouped logically to form groups with a Designated Receiver (DR) as the representative of the local group. These local groupings of the

receivers lead to the formation of several local multicast trees, which together form the global multicast tree. Hence the sender, receivers and the designated receivers form the three major entities of RMTP. The leaf receivers periodically send status messages to their designated receivers (DRs). DRs in turn send their status messages to the DRs in the higher level and so on until the status message finally reaches the source. Thus there is a hierarchical flow of data in the multicast tree from one level to another level. Lost packets are recovered locally and the DR does retransmissions either through unicast or multicast mechanism. To facilitate error recovery for late joining receivers, the source and the DRs buffer the data packets for the session. RMTP uses two level cache mechanisms with the most recent packets are cached in memory and the rest are cached in the disk. Flow control is based on a combination of rate and window-based control.

Although the protocol is scalable to a large number of receivers, it does not provide an end-to-end congestion control scheme. The sender only gets feedback from its own children (the DRs) about their receiving status. Hence, the sender has little information about the congestion status of the receivers. When congestion occurs at leaf receivers, it may mot be possible for the sender to detect the congestion, especially if the DRs and the leaf receivers do not share the same network path. In this case, the sender will continue to transmit at the same rate, aggravating the existing congestion. RMTP traffic may be completely unresponsive to congestion and may cause congestion collapse. Another drawback of the protocol is the dependence on the network equipment such as the routers. The routers have to be modified substantially. Although the information from each receiver is delivered in order, RMTP does not guarantee causal delivery.

2.3.3 Tree-based Reliable Multicast Protocol (TRAM)

TRAM [9] was developed at Sun Microsystems Labs by Kadansky *et al.*. It was designed to support bulk data transfer with a single sender and multiple receivers.

Unlike the protocols discussed earlier, TRAM uses dynamic trees in its architecture. These are used for local error recovery and aids in the scalability of large number of receivers.

The receivers and the sender of the multicast group interact with each other dynamically to form repair groups. The repair groups are linked together hierarchically to form the multicast tree with the source rooted on top of the tree. A subset of receivers is chosen for the reliable delivery of the data. A special receiver called the repair head is chosen among the subset of receivers, and is responsible for the local error recovery and retransmission processing. These repair heads may be statically or dynamically selected. The repair head caches the data packets temporarily until all the receivers in the subset receive them. The source of the multicast tree also caches the data packet and retransmits it if requested.

TRAM uses a rate-based flow control. The data rate of the sender is dynamically adjusted based on the congestion feedback from the receivers. The repair head also sends control messages like congestion notification to its parent; these propagate until they reach the source. Each member of the tree sends feedback reports to its repair head periodically apart from the congestion notification. This feedback consists of general information or statistics, which also aids in the construction of the tree. TRAM also proposes several optimized tree construction techniques suited for different applications. TRAM proposes several tree management techniques for continually optimizing the repair tree.

Though the source and the repair heads cache the data packets for error recovery, TRAM does not support full-data recovery. There may be cases where the receivers joining late would not be able to recover fully due to the absence of data both at the repair head and the source as their buffers are periodically reclaimed. TRAM is currently implemented in Java and several sample applications Slinger, Bricks, Stock and TreeTest have been developed to test its capabilities.

2.3.4 Multicast TCP (MTCP)

Rhee *et al.* [7] proposed the Multicast TCP (MTCP), a point-to-multipoint protocol mainly developed to provide a detailed congestion control scheme for reliable multicast. The protocol describes a detailed congestion control technique similar to TCP congestion control methods.

MTCP also supports a tree-like hierarchical structure. The receivers are not grouped locally. The multicast group is a hierarchical tree with the sender forming the root of the tree and the receivers forming the other nodes of the tree. The sender multicasts the data to all the receivers, and the latter send acknowledgements to their parents in the tree. The internal nodes are called Service Agents (SAs). The leaf receivers send ACKs to the SAs immediately above them, which in turn send them to the SA in the level above and so on until it reaches the source. The receivers send either a positive acknowledgement (ACK) or a negative acknowledgement (NACK). Received packets are reported in ACKs and missing packets are reported in NACKs. The service agents are responsible for handling the feedback generated by the children and retransmitting lost packets.

MTCP provides several features for congestion control mechanisms. The receivers send a consolidated congestion status report up in hierarchy towards the source. A new concept of relative time delay is introduced to overcome the difficulty of calculating the round-trip time. Flow control is window-based, which allows the sender to control the amount packets it multicasts to the group. It also incorporates a selective acknowledgement scheme at the service agents to prevent independent packet loss from reducing the sender's transmission rate. Each service agent maintains a TCP-like congestion window, which operates in a manner similar to the standard TCP congestion control algorithms [16].

MTCP provides a congestion control scheme similar to the TCP congestion control mechanisms. This is achieved through the hierarchical congestion status

reports sent by the leaf receivers to the SAs above them. Each SA monitors the congestion level of its children by independently maintaining a dynamic congestion window using the ACKs and NACKs received from them. The status reports eventually reach the source, the root of the tree, which regulates its transmission rate based on its summary. MTCP also uses Relative time delay (RTD) concept, which overcomes the difficulty of estimating the round-trip times in tree-based multicast environments. Unlike TCP that uses feedback from a single receiver to estimate the round trip time, MTCP open loop system with multiple receivers. MTCP measures the difference between the clock value taken at the sender when a packet is sent, and the clock value taken at the SA when the corresponding ACK is received. The time difference is called the relative time delay in MTCP.

The protocol is not scalable to a large number of receivers due to its architecture. Also, MTCP does not describe in detail the formation of the multicast tree and its management techniques. MTCP primarily describes a congestion control mechanism for reliable multicasting.

2.3.5 Scalable Reliable Multicast (SRM)

SRM, proposed by Floyd *et al.* [3] is a reliable multicast framework for light-weight sessions and application level framing developed at the Lawrence Berkeley Labs (LBL). Although the framework has been designed for a wide range of applications, it has been primarily prototyped in **wb**, a distributed whiteboard application.

The SRM is not hierarchical and hence the scalability is very limited. Whenever a member generates new data, the data are multicast to the whole group. Each and every member is responsible on its own to detect packet loss and to request retransmission. To prevent the implosion of control packets sent from receivers in a multicast group, it adopts a mechanism similar to the XTP [4], by which the control packets are multicast to the whole group. As with the original data, repair

requests and retransmissions are always multicast to the whole group. The other members that are also missing the same packet hear that request and suppress their own request. This prevents a request implosion. A similar technique is adopted to prevent response implosion. The repair requests are different from the traditional NACK in that they are not addressed to a specific sender and they request data by its unique name. In SRM, each member also multicasts periodic session messages that report the sequence number state for active sources. The session messages in SRM are used to determine the current participants of the session.

From the architectural design of SRM, it can be seen that it is not scalable to a large number of receivers. The error recovery scheme is totally different from the other protocols discussed earlier. Separate mechanisms for the suppression of the request and response implosion have to be provided, that could otherwise be avoided in the proper architectural design. Also, the design of SRM results in a problem called the **crying baby** problem. If a single link connecting a member of the group has a very high error rate, then the member sends out the repair request to the whole multicast group and receives one or more responses. Congestion in one part of the group will lead to this problem.

2.3.6 Log-based Receiver-reliable Multicast (LBRM)

The Log-based Receiver-reliable Multicast protocol [5] is a reliable multicast protocol suited for high performance simulation applications, particularly Distributive Interactive Simulations (DIS). It was developed to suit applications that have requirements such as wide-area data distribution, low latency, and packet loss detection and recovery. Holbrook *et al.* give detailed discussions of various applications of LBRM like traffic reports, file caching, stock quotes dissemination.

In LBRM, a **logging server** provides the reliability by logging all transmitted packets from the source. Any receiver that missed a data packet requests the same

directly from the logging server. The presence of the logging server is the equivalent of any transport protocol buffering the data to serve retransmissions. The protocol is receiver reliable in the sense that each receiving application defines its own reliability requirements. The sender merely sends the data and makes it possible, via the logging server, for the receiver to be able to retrieve the lost packet. The source includes a sequence number in each packet and defines a Maximum Idle Time (MaxIT) bound. The source guarantees that it will transmit a packet at least once every MaxIT interval. Even if the application does not provide the data, the protocol keeps sending some special packets called keep-alive or heartbeat packets that repeat the previous sequence numbers and not the associated data.

LBRM also proposes the concept of distributed logging. The receivers are grouped together to form sites. Each and every site has a secondary logging server apart from the primary logging server located near the multicast source. The presence of the secondary-logging servers makes the error recovery distributed. Each of the secondary logging servers is responsible for the retransmission of data packets at their site. This reduces the end-to-end propagation delay. The secondary-logging servers can in turn recover the missing packets from the primary logging server. The designs of many reliable multicast protocols with local error recovery, including the framework described in this thesis, are based on the architecture of LBRM.

2.3.7 Xpress Transport Protocol (XTP)

The Xpress Transport Protocol [4] is a high-performance transport protocol designed to meet the needs of distributed, real-time, and multimedia systems in both unicast and multicast environments. Although XTP was first established in 1987, it was later modified with several versions.

The important features of XTP are the connection paradigms, control algorithms and the group membership controls. XTP designers chose a unique connection-oriented multicast paradigm whereby XTP sets up a one-to-many simplex connection with a set of receivers. The functionality found in the point-to-point unicast connections are extended in XTP. With the control algorithms, XTP provides its users with the ability to enable or disable the error-rate and flow-control procedures on a connection by connection basis. XTP designers have experimented with two methods for control algorithms: a heuristic algorithm for timer-based processing of control information and explicit processing of the state information from each receiver in the multicast group.

XTP is principally a sender-based reliable protocol and it has the option of both unicast and multicast. Control packets are unicast to the sender and there is no mechanism to prevent control packet implosion. Retransmitted packets are multicast and filtered at the sites. XTP supports a number of multicast group management techniques. Though it has support for flow and error control, it uses the mechanisms defined for XTP unicast.

2.4 TCP Congestion Control

This section describes the TCP congestion control mechanisms that have been standardized and researched widely. Earlier implementations of the TCP/IP implemented the go-back-n model without the presence of any modern congestion control algorithms. They used the cumulative acknowledgement mechanism for acknowledging the packets and the re-transmit timer expiration for sending the packets again when they are lost in the network. These implementations did not help in reducing the congestion very much. Van Jacobson, one of the greatest pioneers of congestion control in the Internet, proposed a series of algorithms for congestion control in TCP that eventually became standardized. The algorithms are discussed briefly as follows:

Slow start. The slow start algorithm uses a new variable called congestion window(cwnd). It operates by observing that the rate at which the new packets should be inserted into the network is the rate at which the acknowledgements are received from the other end. The sender can only send in a minimum of the cwnd and receiver's advertised window (rwnd). For each of the ACK the sender receives, the cwnd is increased by one segment. Increasing the cwnd by one for every ACK, results in an exponential increase of cwnd over round trips.

Congestion avoidance. The congestion avoidance uses another variable called a slow start threshold (ssthresh). It indicates the correct window size depending upon the network load. Initially the slow start phase begins. As long as the cwnd is less than the ssthresh, the slow start continues. Once the cwnd crosses the ssthresh, "congestion avoidance" phase starts. For each of the ACK received, the cwnd is increased by 1/cwnd segments. When the sender times out waiting for an ACK, ssthresh is set to a minimum of cwnd/2 and the receivers advertised window. The cwnd is again set to one and the slow start phase begins again. The slow start will be active as long as the cwnd is less than the ssthresh. Else, congestion avoidance phase begins.

Fast retransmission. When the TCP receiver receives an out of order segment, it immediately sends an duplicate acknowledgement. The duplicate ACK indicates the next segment the receiver is expecting and asking the sender to transmit the same. The duplicate ACK may be sent due to the loss of a segment at the receiver's side or due to re-ordering of the segments. If the duplicate ACK was due to the re-ordering of segments, then there may be only one or two duplicate ACKs. If there are 3 or more duplicate ACKs, then it must be due to the missing segment and the sender retransmits the missing segments without waiting for the retransmit timer to go off. TCP then reduces the ssthresh to cwnd/2 and resets the cwnd to one segment.

Fast recovery. The fast recovery algorithm prevents the communication path from going empty after fast retransmit. Therefore, there is no need to slow start from the beginning after the fast retransmit. Fast recovery keeps track of the number of duplicate acknowledgements and tries to estimate the amount of outstanding data in the network. It will increase the cwnd by one segment for each duplicate acknowledgment received, thus maintaining the flow of traffic. The sender comes out of the fast recovery when it receives an ACK for the segment whose loss resulted in the duplicate acknowledgements. TCP will now deflate the window by returning it to the ssthresh and enters the congestion avoidance phase.

Slow start, congestion avoidance, and fast retransmission together form TCP Tahoe and TCP Reno includes fast recovery mechanism.

2.5 Existing QoS Approaches

The Internet today provides only the best-effort service. As discussed in the introduction section, with the growing demand for real-time and non real-time multicast applications, the demand for the quality of service is greatly increased. Although the reliable transport protocol discussed in this document aims at providing reliability, it would be even better if the network layer were to combine with the transport protocol to enhance the same.

The IETF has proposed a number of mechanisms to meet the demand for QoS. The popular ones are the Integrated Services/RSVP, Differentiated Services, Multiprotocol Label Switching (MPLS), traffic engineering and constraint-based routing [17]. Here, the QoS model of Differentiated Services is discussed in detail of how it is incorporated.

Integrated services/RSVP. The Integrated Services/RSVP model proposes two service classes in addition to the best-effort service: Guaranteed Service and Controlled-load service. The RSVP (Resource ReSerVation Protocol) is a signaling protocol that reserves resources in the network for flows. Routers in the network

use the PATH and RESV messages to reserve resources. Differentiated Services, MPLS and other QoS mechanisms are replacing the once-popular Integrated Services, largely due to the following reasons. Integrated Services places a huge burden on the routers due to the amount of state information and processing capabilities it requires. High storage and processing overhead are the major cons involved.

Differentiated services. Unlike RSVP, there is no signaling mechanism in Differentiated Services, thus eliminating the QoS setup costs. The QoS requirements are obtained by modifying the TOS (Type of Service) field in the IP header to a field called Differentiated Services (DS) field. The elements of the DS field are shown in Figure 2.1. Applications make use of the DS field to mark the packets according to their requirements. It is the job of the DiffServ architecture to deliver the packets to the receiver application. Different multicast applications have different requirements such as time or delay sensitivity. Hence they mark the DS packets as per their requirements and the DiffServ architecture takes care of the operation.

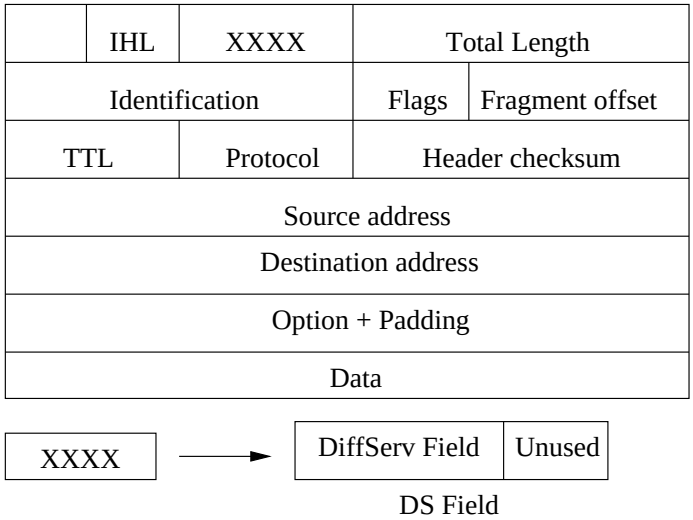

Figure 2.1: Differentiated Service (DS) Field for TOS in IP Header

The DiffServ architecture consists of entities like routers that perform the following functions: classifying, metering, shaping and dropping. The

customer/client should have an agreement called the Service Level Agreement (SLA) with the Internet Service Provider (ISP) to receive the Differentiated Services. Customers mark the DS fields of individual packets to indicate the desired service. Edge routers of the network classify, police and shape the packets according to the SLA. Core routers just forward the packets based on the marking of the DS field. The various functions done by the router are discussed below.

- *Classifying*: The edge router looks at each packet and identifies the flow to which it belongs. Looking at the DS field does this.

- *Metering*: After classifying the flow, its resource consumption should be measured. Measuring the traffic is also important for billing information.

- *Shaping*: The flow may occasionally include some bursts that must be absorbed and the packets must be paced. It is up to the router to decide on the mechanism to hold the burst. It may hold or even drop the packets in the burst that exceed the particular threshold.

- *Dropping*: Whenever the flow exceeds the SLA, a router may choose to drop the packets. Depending upon the type of service subscribed, Assured Service or Premium Service, the packets are handled accordingly. The various queue management schemes like Random Early Detection (RED) and Random Early Detection with Input and Output (RIO) are employed for dropping the packets. In RED, the packets are dropped randomly. RIO is a modified RED algorithm. RIO maintains two RED algorithms, one for the in packets and another for the out packets. There are two thresholds for the queue. When the packets are below the first threshold, no packets are dropped. When the queue size is between the two thresholds, only the out packets are dropped randomly. In extreme congestion, when the queue size exceeds the second threshold both the out and in packets are dropped randomly.

Hence the multicast applications could specify the level of reliability required through the marking of the DS fields. In addition to the transport level reliability provided by the protocol, enhanced reliability could be provided by the network layer using the Differentiated Services as discussed above.

2.6 IETF Approaches

Internet Society (ISOC) is a global not-for-profit membership organization founded in 1991 to provide leadership in Internet related standards, education, and

policy development. The Internet Engineering Task Force (IETF) is an organization which is maintained by ISOC. The IETF is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. The IETF is divided into various working groups (WG) that work on different areas of the Internet working towards its standardization. Although a number of reliable multicast transport protocols have been developed, most of them are used in research while some of them have been released as commercial products. None of the reliable multicast transport protocols have been standardized.

The Reliable Multicast Transport, rmt-working group is chartered to standardize reliable multicast transport protocols for "one-to-many bulk data" transport. This group works closely with other IETF groups such as Secure Multicast research group (smug) and Multicast Security working group (msec). The rmt working group is currently working on the following three protocol instantiations:

- *NORM*: Nack Oriented Reliable Multicast protocol, which uses NACKs for reliability;
- *TRACK*: TRee ACKnowledgement-based protocol, which uses a tree structure for controlling feedback and repairs;
- *ALC*: Asynchronous Layered Coding, which uses Forward Error Correction (FEC) techniques and does not require any feedback.

A number of Internet drafts and RFCs have already been published in the above mentioned areas. The much-researched areas in reliable multicasting by IETF include congestion/flow control measures, efficient retransmission, and acknowledgement aggregation. There are numerous publications in the literature on the same.

IETF has framed some technical criteria [18] for Reliable Multicast transport protocols. The criteria include addressing the following key issues:

- *Scalability*: The ability of the protocol to accommodate a large number of senders and receivers and the mechanisms that limit the scalability;
- *Congestion Control*: Description of the congestion control mechanisms that are incorporated and their behavior during congestion;
- *Error Recovery and Robustness*: Description of how the protocol handles the packet loss and node/link failures;
- *Security and Privacy concerns*: Analysis of security at the senders, receivers, routers and retransmission sources along with data integrity and authentication.

The framework proposed in this thesis addresses all of the issues mentioned above except the security aspects. The protocol is scalable to a large number of receivers by its architectural design. Logical grouping of receivers localizes the error recovery and it also mentions a congestion control scheme that operates on congestion algorithms similar to TCP. The security issues are just mentioned and not discussed in detail.

2.7 Summary

Due to the strong application demand of reliable multicasting, there is a huge interest in the area of standardization of the same. Therefore it continues to be an active research area in the Internet community.

This chapter discussed the various multicasting techniques and their applications. Reliable multicast protocols may be broadly classified as sender-initiated or receiver-initiated protocols. The chapter also reviewed the various reliable multicast transport protocols in the literature.

LGMP, RMTP and TRAM protocols share the technique of local grouping of receivers in their designs. The local error recovery design of most of the reliable multicast protocols was inspired by the design of LBRM. Although LBRM proposes a distributed logging system with the receivers grouped as a site, it does not mention any of the group management techniques as in TRAM. RMTP groups the

receivers locally leading to the formation of several local multicast trees, which together form a global multicast tree. Group management techniques are not discussed in detail in RMTP either. The local groups of receivers have a special receiver called repair head, responsible for retransmissions and recovery. TRAM is rich in group management schemes and has techniques for continually optimizing the multicast tree structure. MTCP was developed primarily to implement a congestion control strategy for reliable multicasting. XTP, the sender-based protocol was developed aiming at distributed, real-time and multimedia systems. SRM is not hierarchical and does not provide logical groupings either. The architectural design of SRM makes its scalability very limited.

The next chapter discusses the various issues in reliable multicasting, which make it difficult for the standardization process. In Chapter 4, the framework of the protocol is proposed.

CHAPTER 3
ISSUES IN RELIABLE MULTICASTING

Design of a reliable multicast protocol is very complex when compared to the design of unicast communication protocols. There are several issues to be dealt with when dealing with group communication. Listed below are some of the important problems encountered with reliable multicasting.

3.1 ACK Implosion problem

This is most the important problem to be dealt with the design of a reliable multicast transport protocol. The sender receives feedback or acknowledgements from the receiver to update and control its regulation parameter for reliability and congestion control. In unicast communication, the sender receives its feedback just from the single receiver present. In the case of multicasting, the feedback messages received from the multiple receivers may converse on the source. This is called the *feedback implosion problem*. There are a few ways to deal with this problem. Some of the existing protocols propose negative acknowledgement (NACK) to be sent to the sender, instead of a positive ACK, so that the sender receives only a limited number of responses from the receivers. If the multicast tree is large and the error probability is relatively high, there would be many losses and the NACKs sent would still cause implosion. Hence there has to be some way of suppressing the NACKs too. Some of the ways to deal with the implosion problem is to use the local grouping scheme and some suppression techniques. Figure 3.1 depicts typical case of the ACK implosion problem. The implosion at the source could also be caused by the NACKs. Hence it is called in general as the feedback implosion problem rather than the ACK implosion problem.

LGMP, RMTP and TRAM discussed in the previous section group the receivers to form small local groups. The receivers in the local group send their ACKs to the entity that controls the group and not to the sender. Hence the sender of the group is not bombarded by ACKs when the data are received properly. In all the three protocols, the ACK implosion problem is avoided by the design of the protocol. MTCP has a hierarchical structure with the receivers arranged in a tree-like fashion. There is no grouping concept in MTCP and the receivers send their ACK to their immediate parent. The parents send their ACK to their corresponding parent and so on. Hence, the source will get the ACK only from the level one receivers and this avoids the ACK implosion problem. SRM does not provide grouping or any hierarchical structure for the receivers. SRM does not ACK the data packets but the NACK packets are multicast to everyone. Since NACK is sent only for a missing packet, it does not always lead to implosion problem. In LBRM too, no ACK packets are sent to the source. LBRM sends the NACK packets specifically to the source. XTP unicasts the control packets to the source and there is no mechanism to avoid the implosion at source.

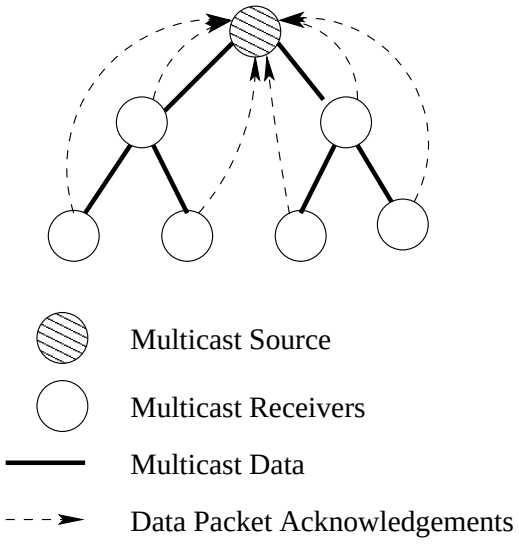

Figure 3.1: ACK Implosion at Multicast Source

3.2 Error Recovery

There are usually two types of error recovery [19]: **centralized error recovery (CER)** and **distributed error recovery (DER)**. In CER the retransmissions are performed by the source of the multicast tree. CER is also referred as source-based recovery. In DER, all the members of the multicast perform the retransmissions group. Hence the burden of the error recovery processing is distributed from the source to all the members of the multicast tree. DER is found to outperform the CER, because the source may not always have sufficient processing power or buffer space to support error recovery, especially when the number of receivers is very large, which is typical of most of the multicast applications. Also for reasons such as fault tolerance, distributed error recovery is recommended compared to centralized error recovery [19]. Kasera *et al.* have shown that local recovery has the potential to provide significant performance gains in terms of reduced bandwidth and delay, and higher throughput [20].

LGMP, TRAM and RMTP provide a distributed error recovery mechanism through the local grouping of receivers. In LGMP, any member of the group performs retransmissions and the packet is requested from the source only when none of the members have it. In TRAM, the repair head performs the retransmissions to the group members whereas a designated receiver is responsible for error recovery in RMTP. MTCP has a hierarchical structure of receivers with each internal node called as Service Agent (SA). SAs are responsible for the error recovery and it requests a packet from its parent if it does not have one. In SRM, the repair or retransmission requests are multicast to the whole group. All the members hear the request and any other member that is missing the same packet would suppress its request on hearing the same request by another member. Retransmissions are also multicast to the whole group and on looking at the response other members would suppress their response. LBRM proposes two types

of error recovery. The protocol has a logging server adjacent to the sender and all the receivers request missing packets from the logging server. LBRM also provides distributed logging by grouping a set of receivers called as "site". Every site has a secondary logging server that provides the error recovery for the receivers in the particular site. Hence, in distributed logging, the recovery becomes localized. In XTP, the repair requests and retransmissions are multicast to the whole group similar to SRM.

3.3 Congestion Control

Congestion control mechanisms in multicasting remain one of the most widely researched areas. Unlike the unicast mechanism, in multicast, where multiple receivers are involved, effective congestion control relies on accurate and timely feedback on the prevalent network conditions. The challenge lies in how economically, speedily and accurately is the feedback information collected.

Golestani and Sabnani [21] have discussed in detail the various issues to be considered by a multicast congestion control scheme. The two major components in the congestion control structure are the regulation parameter and the regulation algorithm. A regulation parameter is a parameter by which the flow of traffic onto the network is regulated, which may be either the rate at which the data is transmitted or the window size. A regulation algorithm is the algorithm by which the regulation parameter is adjusted. In the standard TCP congestion control mechanism, the regulation parameter is the window size and the regulation algorithm is the combination of the standard congestion control algorithms which were proposed by Van Jacobson, like Slow start, congestion avoidance, etc. Most of the congestion control schemes today adopt the regulation parameter as either the transmission rate or the window size. There are difficulties in extending window-based regulation to multicast communications because of the concerns like ACK implosion problems. The major drawback in implementing rate-based

regulation is the necessity to calculate the receiver round trip times. The measurement of receiver round trip times is fundamentally different and more complex than the unicast communications. In unicast communications, where there is a single receiver, the round trip time is measured easily. In the multicast communication with multiple receivers involved, issues arise of measuring for each receiver. In multicast protocols with logical groupings, the receivers must measure their distances from the leader of the group.

The main difference between the congestion control schemes implemented in unicast and multicast communication is the place at which the regulation algorithm is run. Accordingly congestion control scheme may be categorized as source-driven or receiver-driven. In source-driven unicast communication schemes, the task of updating the regulation parameter is left to the source. The source upon getting the feedback from the receiver is the ideal site to run the regulation algorithm. In multicast communication, Golestani [21] discusses the following problems.

- In multicast sessions, due to the presence of a large number of receivers, the complexity of performing the traffic regulation is also increased. Unlike the unicast methodology, if the execution of the regulation algorithm is left solely to the source, its processing capability could severely limit the same.

- In the protocol discussed in this thesis, many retransmissions are not performed by the source, rather they are being performed by the Group Leader. Hence the loss information pertaining to the various receivers is not always available to the source.

- In the current hierarchical local grouping architecture, the number and identity of the receivers is not known to the source.

- The congestion control decisions are not always implemented by the source. If a receiver decides to drop out of the group, the receiver alone must implement it.

Another important factor to be taken into account is the fact that the natural approach for congestion control is to adopt TCP's congestion control algorithms for

reacting to network congestion. Floyd *et al.* [22] have proposed a guideline as follows.

> For any link, the traffic arrival rates for a flow should respond to congestion in a way that is no more aggressive than multiplicative decrease, additive increase, with increase and decrease rates that give behavior that is no more aggressive than current implementations of TCP.

Considering the above-mentioned factors, the real need of the congestion control in reliable multicast is to shift the tasks as much as possible to the receivers. It should be a receiver-driven approach with the receivers sending the feedback about congestion to the source. The control algorithms should also be TCP-friendly for reasons as discussed above.

The architectural design of the multicast protocol has a major effect on the design of the algorithms that aid in sending the feedback from receivers to the sender. Golestani and Sabnani have proposed a hierarchical consolidation of receiver feedback in the multicast tree. Each receiver is responsible for consolidating the feedback received from its immediate children and sending the result upward to its parent. At each level inside the tree, the receiver computes an aggregate feedback parameter at its level based on the feedback received from the level below it. The computed aggregate feedback parameter is then sent upward towards its parent until it reaches the source finally.

LGMP does not mention any congestion control mechanisms. TRAM specifies ways in which the receivers in the group send notification about congestion to the source. Upon the reception of the congestion notification, the source reduces its transmission rate. TRAM specifies a minimum and maximum transmission rate within which the protocol operation takes place. On reception of congestion reports, TRAM reduces its transmission rate by 50

3.4 Scalability

With the multimedia applications growing tremendously, the number of receivers in the multicast groups can be expected to increase proportionately. Hence the protocol should be designed in such a way that it can handle the increase in number of receivers and still provide the same level of service. Also, the multicast receivers should be able to join or leave the group whenever they wish and the source should not be even aware of these activities. The architectural design of the protocol plays a major role in deciding the scalability of the protocol.

LGMP, TRAM and RMTP combine a set of receivers to form local groups. Due to the logical grouping of receivers, the protocol is scalable to a large number of receivers. MTCP does not provide any grouping of receivers, but the receivers form a hierarchical tree-like structure with the sender rooted on top of the tree. The scalability of MTCP is limited due to its hierarchical nature. SRM provides neither grouping nor hierarchical structure for the receivers. The receivers just form one group under the sender which limits its scalability very much. SRM was mainly developed for "white board" application where the number of receivers is limited. Distributed logging in LBRM is scalable to a large number of receivers as they are grouped together to form "sites". XTP does not provide any grouping of receivers and its scalability is limited.

3.5 Fairness

A particular concern for the developers of reliable multicast protocols is the impact of reliable multicast traffic on other traffic in the Internet at times of congestion, in particular the effect of reliable multicast traffic competing with TCP traffic [18]. The protocol should address the fairness issue. There are many possible ways to define fairness. One type of fairness is global fairness. Under this definition, each entity has an equal claim to the network's scarce resources. The multicast protocol should ensure fair sharing of network resources with other

well-behaved protocols. It should ensure fairness with other multicast and unicast traffic. The multicast protocol should behave and back off in a way similar to TCP in case of congestion. This is called TCP friendliness. Hence the reliable multicast protocol should be designed in such a way that it is fair.

Protocol	ACK Implosion	Error Recovery	Congestion Control	Scalability
LGMP	✓	DER	×	✓
TRAM	✓	DER	✓	✓
RMTP	✓	DER	×	✓
MTCP	✓	DER	✓	×
SRM	×	DER	×	×
LBRM (Primary)	×	CER	×	×
LBRM (Secondary)	✓	DER	×	✓
XTP	×	DER	×	×
TCP	—	✓	✓	—

✓ Handles Well × Suffers from this — Not Applicable

DER − Distributed Error Recovery CER − Central Error Recovery

Figure 3.2: Comparative Analysis of Reliable Multicast Protocols

3.6 Summary

This chapter discusses various important issues involved in reliable multicasting. The implosion problem is a primary problem to be addressed in a reliable multicast protocol. The acknowledgement sent by multiple receivers should not bombard the sender. The architectural design of the protocol has a major impact upon the implosion problem. The error recovery protocol could be centralized or distributed.

Distributed error recovery reduces the load of sender and provides efficient retransmission mechanisms. The reliable multicast protocol should respond to the congestion state in the network through congestion control algorithms. The multicast protocol should also be scalable to a large number of receivers.

From the above issues it could be seen that design of a reliable multicast protocol is complex when compared to the design of unicast protocols. The architectural design of the protocol plays an important role and has compounding effects on issues such as implosion problem, error recovery and scalability. This chapter also evaluated the various reliable multicast protocols in literature to the issues such as implosion problem, error recovery, congestion control and scalability. Figure 3.2 shows the comparative analysis of LGMP, TRAM, RMTP, MTCP, SRM, LBRM, and XTP. The analysis also depicts the relation of TCP with the reliable multicast protocols.

CHAPTER 4
PROTOCOL FRAMEWORK

4.1 Goals

4.1.1 ACK Handling

One of the major design decisions of any reliable multicast protocol is how to overcome the ACK implosion problem. The protocol framework discussed in this thesis has an architecture designed in a way that will avoid the problem automatically. The multicast receivers are grouped locally with a local leader called the Group Leader (GL) for each particular group who is in charge of the error recovery and local retransmissions. Instead of the receivers sending positive ACKs back to the source, they only send NACKs to the Group Leader whenever they detect the loss of a packet. Hence the source is never flooded or imploded with NACKs either. All they receive is the ACKs from the Group Leaders. This way, the ACK implosion problem is mitigated.

4.1.2 Error Recovery

As the multicast receivers are grouped locally, error recovery is taken care by the Group Leader which makes it Distributed Error Recovery (DER) [19]. The Group Leader locally handles any re-transmissions. Hence the error recovery time is minimized greatly, as the re-transmission requests do not have to propagate all the way up to the source every time a packet is lost. This results in speedy recovery and distribution of the processing throughout the multicast tree. The end-to-end delay of the retransmission request and repair is reduced.

4.1.3 Congestion Control

Many of the earlier protocol implementations for reliable multicasting do not specifically address the congestion control techniques. The protocol framework

proposed in this thesis follows the TCP congestion control algorithms closely. The algorithms are implemented and handled locally in the groups by the Group Leader. The Group Leaders maintain a TCP-like congestion window (cwnd). Group Leader increments the cwnd by one only when it receives an ACK from each of its children. Until then, the Group Leader has to buffer the packets. As soon as the ACKs are received, the buffers are released. The algorithms are similar to the standard TCP congestion control algorithms like slow start and congestion avoidance. Details of the algorithm are specified in Section 4.5. The protocol also proposes a scheme where a congestion feedback is propagated up in the multicast tree hierarchy.

4.1.4 Scalability and Dynamic Adaptation

Since the receivers are grouped locally with a leader acting on their behalf, the protocol is scalable to a large number of receivers. A new receiver who wishes to join in the multicast could join with any of the existing local groups or it could form its own group announcing itself as the Group Leader. Hence the multicast tree grows hierarchically in groups as the receivers increase in number. Also, a receiver may leave its group any time it wishes and all this may happen without the knowledge of the multicast source.

The most important feature of this protocol is the dynamic way in which the members of the group may re-arrange themselves in response to changing network conditions. The Group Leaders exchange special control messages that enable them to do so. Some control messages are multicast to the group too, so that the members of the local groups also have updated information about the state of the multicast group.

4.2 Architecture

The basic architectural design of the protocol is shown in Figure 4.1. The multicast receivers are grouped locally to form small groups called the local groups.

Each group contains a special node called the Group Leader (GL). The main motives behind the local groupings are to increase the scalability of the protocol and to localize error recovery. The Group Leader is responsible for acknowledgement processing and buffering of the packets. They also handle processing of the control messages from the multicast receivers, used for congestion control and group management, and pass these messages up to the multicast source. As the error recovery is localized within each of the local domains, the Group Leader is responsible for local re-transmissions. As it contains the buffered data, it would be able to re-transmit the data requested by any of its children. In all, the Group Leader plays a major role in the functioning of the protocol. In Figure 4.1, the circled dots represent the logical local groups. The buffering at the Group Leaders in only temporary whereas the central logging server adjacent to the source provides a permanent buffering. In addition to multicasting the packet to the group, the source sends the packets to the central logging server, which retains them permanently.

All the local groups together form a tree-like hierarchical structure as shown in Figure 4.1. Hence the source of the multicast tree is directly connected to the Group Leaders and the Group Leaders in turn are connected to other Group Leaders and the multicast receivers. The Group Leaders receive ACKs and NACKs from their children, the multicast receivers in the local group and the Group Leaders attached below, if any. The source receives ACKs and NACKs from its immediate children, which are Group Leaders themselves. Due to this hierarchical structure, the protocol is scalable to a very large number of receivers.

4.3 Multicast Data Transfer

The following sections discuss in detail the various functions of the protocol such as the multicast data transfer, acknowledgement and local retransmission schemes, congestion control and the group management techniques.

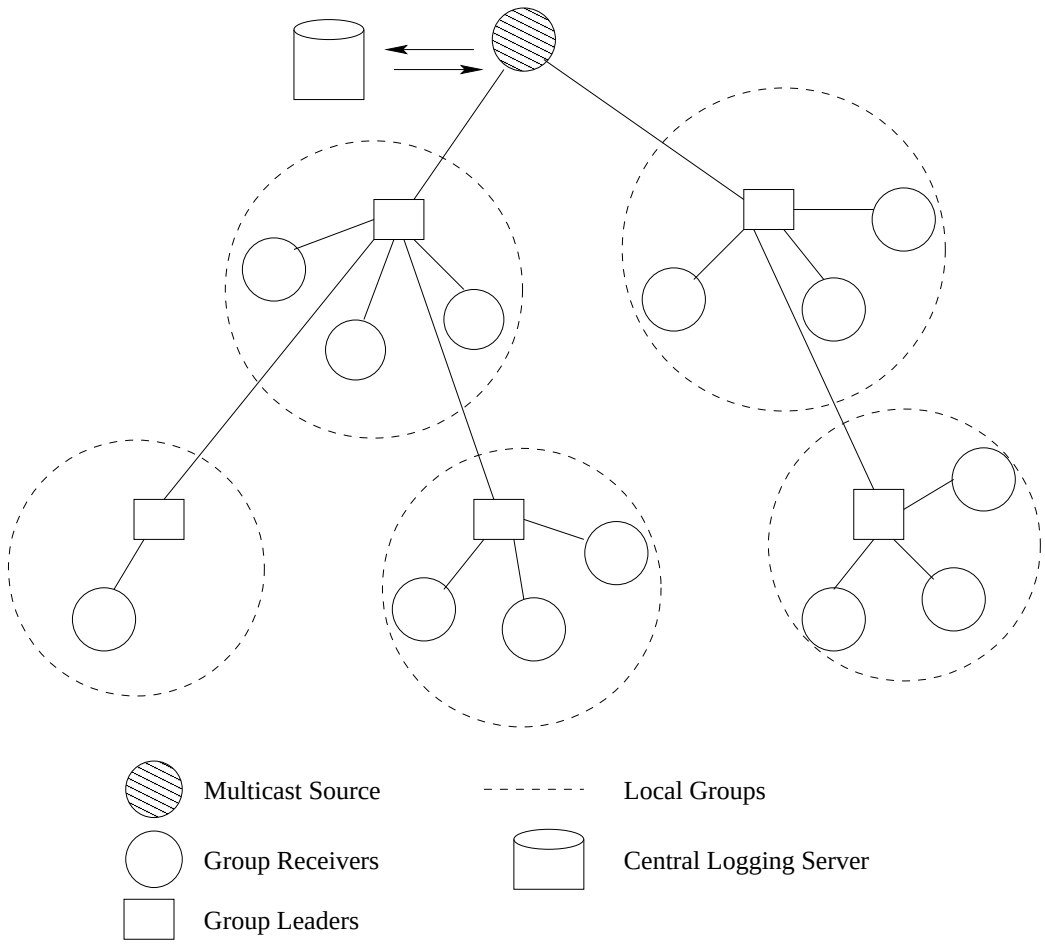

Figure 4.1: Overall Architectural Design of the Protocol

The architecture of the protocol is very much like a hierarchical tree-like structure with local groupings as discussed in the previous section. This would avoid the end-to-end delay of transmission and re-transmission. The multicast data transfer works as follows. The source or the sender first multicasts the data globally to all the members of the multicast tree, including the Group Leaders and the individual receivers. The Group Leaders, that are logically connected to the source send their ACKs and control messages directly to it. Each Group Leader receives ACKs and NACKs from its children, the receivers in the local group and any Group Leaders attached directly below it, if any. The receivers do not send ACKs directly to the source bypassing the Group Leader. Hence the source is never bombarded by ACKs from the receivers, which avoids the ACK implosion problem.

The Group Leader locally will retransmit any missing data units requested. If the Group Leader does not has the data packet, it would request for retransmission from its parent Group Leader. The Group Leader either multicast or unicast the missing data and it also must buffer the packets until it receives an acknowledgement from all of its children. The source multicasts the data unit as long as it has room in its sending window. The flow of data packets and acknowledgements is shown in Figure 4.2.

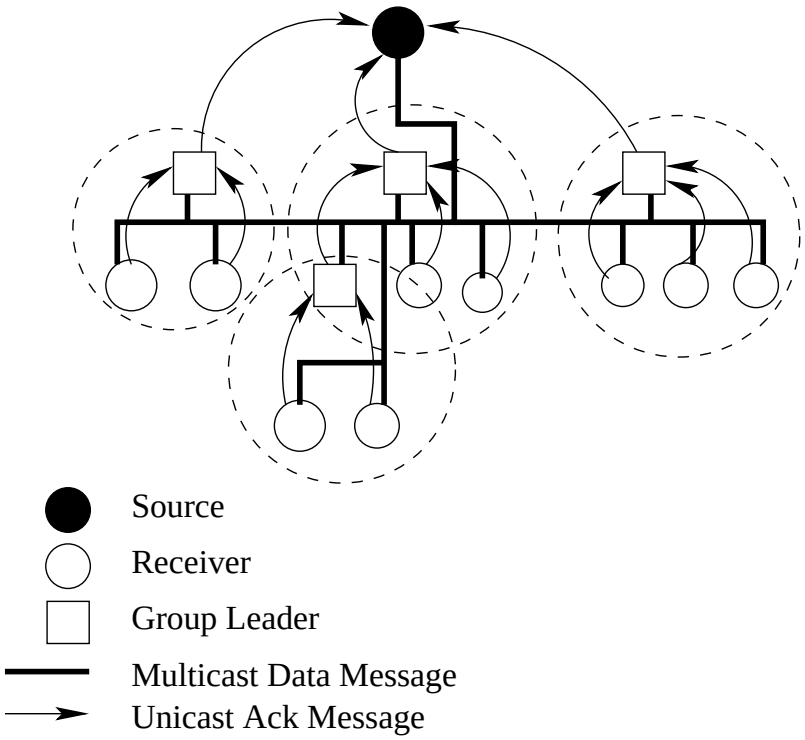

Figure 4.2: Multicast Data Transfer

The multicast data transfer from the source to the multicast group and the whole multicast operation takes place in the following order.

1. Source multicasts the data unit globally to the entire multicast tree, including the Group Leaders, central log server and the receivers.

2. The Group Leaders attached directly to the source would send their ACKs and NACKs to the source.

3. The central log server, which is also attached directly to the source, sends its ACKs and NACKs to it.

4. The Group Leaders receive the ACKs and NACKs from their children which may include the local receivers and any Group Leaders attached directly below them.

5. The source performs re-transmissions to its immediate children, the Group Leaders as well as the central log server.

6. The Group Leaders perform local re-transmissions to their children.

7. The central log server performs any re-transmissions if requested.

8. Steps 1 through 7 are repeated for protocol operation.

4.4 Acknowledgement and Error Recovery Mechanism

In his paper [19], Nonnenmacher has shown that for Distributed Error Recovery protocols, the performance of the protocol is high with a local multicast retransmission and local feedback-processing scheme. As discussed in the previous sections, one of the major problems encountered with reliable multicast design is the ACK implosion problem. This problem is unavoidable in the sender-initiated reliable multicast protocols, wherein all the receivers send their positive ACKs to the sender impounding it virtually.

The protocol discussed in this thesis is a receiver-driven approach with a slight modification. Instead of each receiver maintaining the state information, the Group Leader alone maintains the state information for the local group it coordinates. The acknowledgement scheme of the protocol works as follows.

There are two types of acknowledgements, the positive ACK that indicates the receipt of the packet correctly and NACK, the negative acknowledgement that indicates the absence of a data packet. As mentioned in the multicast data transfer section, the source transmits the data packet to the whole group. After the receipt of the packets, the Group Leaders that are directly attached to the source send their ACKs to the source. Hence the source is not bombarded by ACKs from all the receivers and the implosion problem is avoided. A Group Leader receives the

ACKs from the receivers in its group and also from the other Group Leaders attached directly below it, if any. The protocol framework proposed in this thesis is designed for a sequenced, loss less delivery of data. Missing packets are detected based on the sequence number of the data packet. Both the Group Leaders and the receivers send NACK if they miss any data packets. The Group Leaders and the source buffer the packets to retransmit any missing packets. Thus there is a hierarchy of flow in the multicast tree between the source, Group Leaders and the receivers. In essence, the receivers could not bypass the Group Leaders to reach the source. Unlike the SRM [3], where the NACKs are multicast to everyone, here a receiver sends its NACKs only to its Group Leader.

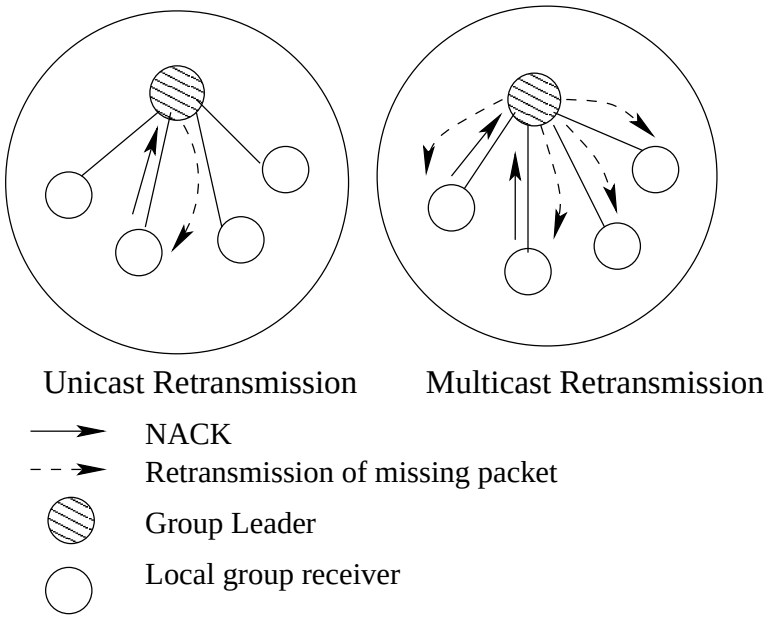

Figure 4.3: Local Retransmission of Missing Packets

The Group Leader and the source receive a NACK for the missing data packets. After the reception of NACK, both Group Leader and source performs retransmissions. The following section deals with the retransmissions in detail.

4.4.1 Retransmission by Group Leader

Retransmission may be done by the Group Leader in two different ways: unicast and multicast. Both the retransmission techniques are depicted in Figure 4.2.

Unicast retransmission mechanism. If the children receive the packets properly, they send an ACK to the Group Leader. The Group Leader releases a packet from the buffer when it receives an ACK for the packet from all of its children. The Group Leader retransmits the missing packet locally to a receiver upon receiving NACK from it. The Group Leader maintains a certain threshold for the number of NACKs it has received from its children for a particular packet. As long as the number of NACKs received for a packet is below the threshold, the Group Leader unicasts the missing packet to the requesting receiver. End-to-end propagation delay is greatly reduced due to the hierarchical structure, since the receiver need not send the retransmission request all the way to the source and wait for the next packet to make its way all the way back from the source, but instead sends its NACK to its Group Leader receives the retransmission from it.

Multicast retransmission mechanism. The Group Leader waits for a certain interval of time before it serves the retransmission request. If the number of retransmission requests for a data packet is found to exceed the threshold, then instead of unicasting the missing packet to the individual receiver, the Group Leader multicasts the packet to the whole group. Hence the local traffic caused by the retransmission requests is reduced by this multicast retransmission mechanism. Because of local recovery, the repair requests are not sent all the way up to the source of the multicast tree. The control traffic is greatly reduced through the local recovery mechanism. When the retransmission request is sent by a Group Leader attached directly below the source, the retransmission is always unicast.

4.4.2 Retransmission by Source

The Group Leaders attached directly to the source send a NACK to the source if they miss any data packets. The source buffers the data packets until it receives an ACK from each of the Group Leaders attached directly to it. The source

retransmits any missing data packet from its buffer when it receives a NACK from a Group Leader attached to it.

The other important concerns of the recovery mechanisms are as follows:

- monitoring of unresponsive receivers and Group Leaders;
- late joining receivers and data recovery.

To ensure reliable delivery of data, the Group Leader buffers the packets until it receives an ACK from each of its children. So it is important for the Group Leaders to monitor the receivers in its local group continuously. If some of the receivers become unresponsive, the Group Leader will not get an ACK for a particular packet and it remains in the buffer infinitely. Therefore it is up to the Group Leaders to monitor the receivers. Also there is a chance that the Group Leader too may become unresponsive sometimes. In that case, the receivers could not get their retransmissions from the Group Leader and would have to get them from another Group Leader, the sender or the central logger. The monitoring mechanisms adopted by the Group Leaders and receivers are discussed next.

4.4.3 Monitoring of Unresponsive Receivers and Group Leaders

Monitoring for unresponsive receivers and Group Leaders is achieved using a control packet called GL_ALIVE, a Group Leader alive message. The Group Leader sends the GL_ALIVE packet periodically to its local group. Reception of the GL_ALIVE message by the local receivers indicates that the Group Leader is operational. If a receiver does not receive the GL_ALIVE packet for a certain interval, it indicates the absence of the GL_ALIVE in the ACK packet it sends to the Group Leader. If three such ACK messages go unanswered, the receiver infers that the Group Leader is down and it joins with a different Group Leader. The methods by which the receivers subscribe to a different Group Leader are discussed later in the group management section. If the Group Leader receives an ACK packet indicating the non-reception of GL_ALIVE, it unicasts a GL_ALIVE to the

requesting receiver. After the reception of the unicasted GL_ALIVE message, the receiver sends an ACK indicating its reception and remains attached to the Group Leader and continues its regular operation.

The Group Leader expects an ACK message within a certain interval from each receiver. If the Group Leader does not receive an ACK message from a receiver for three such intervals, the Group Leader then unicasts a GL_ALIVE message to the specific receiver indicating the absence of the ACK message. If the Group Leader does not get a response for the unicasted GL_ALIVE message, it assumes that the receiver is down and it releases the data in the buffer it was withholding for that receiver. Also, the Group Leader presumes that particular member is down and no further repairs would be served for it. The member has to join again either with the same Group Leader or a different one if it wishes in the future to participate in the multicast operation.

4.4.4 Late Joining Receivers and Data Recovery

Since the protocol discussed in this thesis allows receivers to join anytime during the multicast session, the receivers joining the group late have to be updated with the data packets sent since the start of the session and catch with the rest of the group members. Also because of the highly dynamic nature of the protocol, the receivers in the local group could change their membership to different groups. This may require them to catch up with the current group data reception. There are two ways in which this could be achieved: (1) from the buffer at the Group Leader or the source and (2) directly from the central logging server for complete data recovery.

Buffer at the group leader and source. The Group Leader buffers each data packet sent by the source until it receives ACKs from all of its children for that packet, after which it deletes the buffer entry for the particular packet. The newly joined member could recover the data from the buffer withheld by the Group

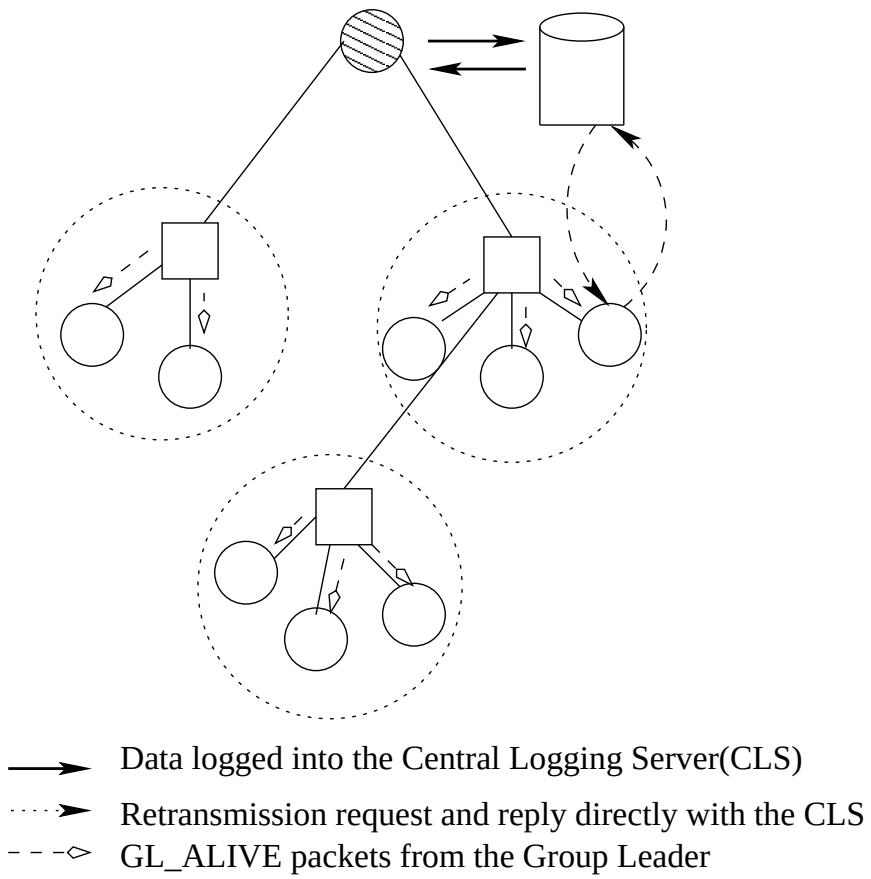

Figure 4.4: Central Logging Server and Monitoring of Unresponsive Members

Leader by requesting it. If the Group Leader has deleted the data from buffer needed by the new member, only partial recovery is possible. Hence it is the duty of the joining member to ensure that it has all the packets that have been acknowledged by all the members of the joining group. If the new member has not yet received some of these packets, it is its responsibility to finish all the pending transactions with the old Group Leader. Alternatively, if the new joining member cannot find the data in the new Group Leader's buffer, it can request the same directly from the central logging server, discussed as follows.

Complete data recovery from central logging server. There is a central logging server attached to the source of the multicast group as in Figure 4.4. It buffers the data on the disk as the source multicasts the data. Unlike the data buffered by the source or the Group Leaders, the data in the central logging server is never deleted.

Therefore any late joining member, that cannot recover fully with the Group Leader's buffer, can request the packets directly from the central logging server. Hence complete recovery is possible. The central logging server also serves as a fault tolerant mechanism for the entire multicast session by buffering the data permanently.

4.5 Congestion Control Mechanism

Section 3.3 discussed the challenging factors involved in the design of congestion control in multicast protocol. The congestion control for the protocol framework is designed as follows. There are two phases involved in the whole process: *slow start* and *congestion control*. In the *slow start* phase, the protocol tries to find an appropriate operating point. As the packet flow gradually increases and when the network is subjected to load, the *congestion control* phase starts to operate. Both the topics are discussed in detail below.

4.5.1 Slow Start

The approach taken in this scheme is similar to and motivated by the congestion control strategy used in MTCP [7]. The source uses a congestion window *cwnd* to reduce the data transmission rate when experiencing congestion. Each Group Leader too maintains a congestion window, similar to the TCP congestion window. The source and the Group Leaders maintain their congestion windows using TCP congestion control mechanisms such as slow start and congestion avoidance. The manner in which the congestion algorithms differ is that a congestion window present at the Group Leader or the source is incremented only when it receives ACKs from all its children. Hence the window size is incremented linearly as and when the ACKs are received from the children. This operation is continued as long as the congestion window size is below the slow start threshold. If the size exceeds the threshold, the congestion window is reduced to 1/cwnd each time a new packet is acknowledged by all of its children and the protocol enters into the congestion

avoidance phase. Also, as discussed in the previous section, the local receivers send NACKs for the missing packets and the Group Leaders immediately retransmit the packets reported missing. It could be seen that the slow start phase discussed for this protocol is very much similar to Van Jacobson's [16] standard congestion control algorithm.

4.5.2 Congestion Control

After the initial phase of slow start, congestion could occur and would be detected by the receivers, the Group Leaders, or the source itself. Accordingly, the receivers and the Group Leaders report the congestion through congestion reports up the hierarchy so that the source acts to reduce the same. The strategy adopted is similar to and motivated by the congestion control scheme used in TRAM [9].

Congestion at receivers. Receivers detect the congestion according to the missing packets. If the receivers detect the number of missing packets in an ACK window to grow beyond a certain threshold, it sends a congestion message to its Group Leader. The congestion message would include the highest sequence number of the packet received. Upon receiving the congestion message, the Group Leader in turn forwards it to the Group Leader above it or to the source if it is attached to the source directly.

Congestion at group leaders. Group Leaders detect congestion when their buffer begins to fill up. The buffer space is filled up when a receiver in the local group fails to acknowledge data packets. The buffer gets filled up with the packets as long as it does not receive the acknowledgements from the receivers. The buffer has a high threshold limit and the Group Leader temporarily increases the high threshold limit set for the buffer when it could not remove any more packets from it. Meanwhile, a congestion message is sent up the hierarchy indicating congestion. If the buffer is filled even after the high threshold is increased and the buffer size is reached, the Group Leader starts to drop the new packets. The temporary increase

of the threshold limit of the buffer is to allow some time for the Group Leader to send a congestion message up the hierarchy.

Congestion at source. The source of the multicast group too maintains a buffer to retransmit any missing packets for the Group Leaders attached immediately to it. Like the above-mentioned situation, the source would detect congestion if its buffer begins to fill up when a Group Leader fails to acknowledge data packets. Similar to the Group Leaders, the source would increase the high threshold limit for its buffer. Meanwhile it reacts to the congestion by reducing the data rate, as if it had received a congestion message. After the high limit of the threshold is reached and the buffer is full, it blocks any new data from the application and attempts to solicit an ACK from the Group Leaders that are causing the buffer to fill up. If the Group Leaders do not respond quickly, the source would prune them. There is a significant policy choice here as to prune the slow receivers or to catch up in speed with the slow receivers. If there is just one receiver that affects the rate of multicast group, it is pruned and if there are a number of receivers that dictate the rate, the sender is slowed down.

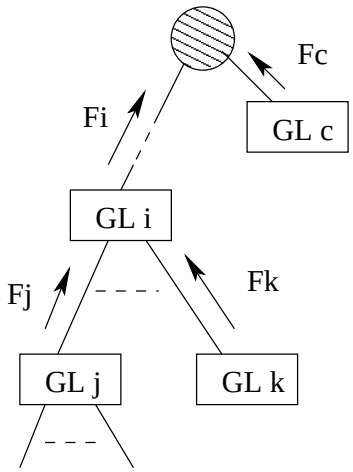

Figure 4.5: Hierarchical Consolidation of Feedback Parameter

Apart from the congestion report sent by the receivers and Group Leaders when encountering congestion, the protocol also sends consolidated congestion control

feedback to the source of the multicast group at regular intervals so that it can regulate the flow of transmission accordingly. Section 3.3 discussed about the hierarchical consolidation of the feedback from the receivers up the multicast tree proposed by Golestani and Sabnani [21]. Every receiver calculates an aggregate feedback parameter based on the feedback received from the receivers at the lower level. This technique is modified to suit the requirements of this protocol framework. It is the job of every Group Leader to calculate the consolidated feedback of its children and the Group Leaders below it. After calculating the consolidated feedback at its level, it sends the same to the Group Leader above its level. If a particular Group Leader does not have another leader attached below (leaf node), it just passes its feedback up the tree. Also, if a Group Leader is independently attached to the source of the tree, it sends the feedback directly to the source. This is depicted in Figure 4.5.

The feedback parameter f_j denotes the highest packet sequence number that could arrive at j in the case of window based congestion control. If f_j is the feedback parameter of a node N_j, then the consolidated feedback at the current level is calculated as follows.

$$\mathrm{f}_j = \min\{\mathrm{f}_k/\mathrm{N}_k, \text{ where } \mathrm{N}_k \text{ is a child of } \mathrm{N}_j\}$$

The consolidated feedback at the current level f_j is then propagated to the level above it, if any or to the source directly.

4.6 Group Management Schemes

This section deals with the various group management schemes involved in the protocol operation. The architecture of the protocol is a hierarchical tree-like structure with local group formation. This necessitates effective organization and management of the groups. Issues such as formation of the group, dynamic configuration of the members and group termination are discussed as follows.

4.6.1 Group Formation

There are two ways in which the local groups are formed: Expanding ring structure [6] and the advertisement method.

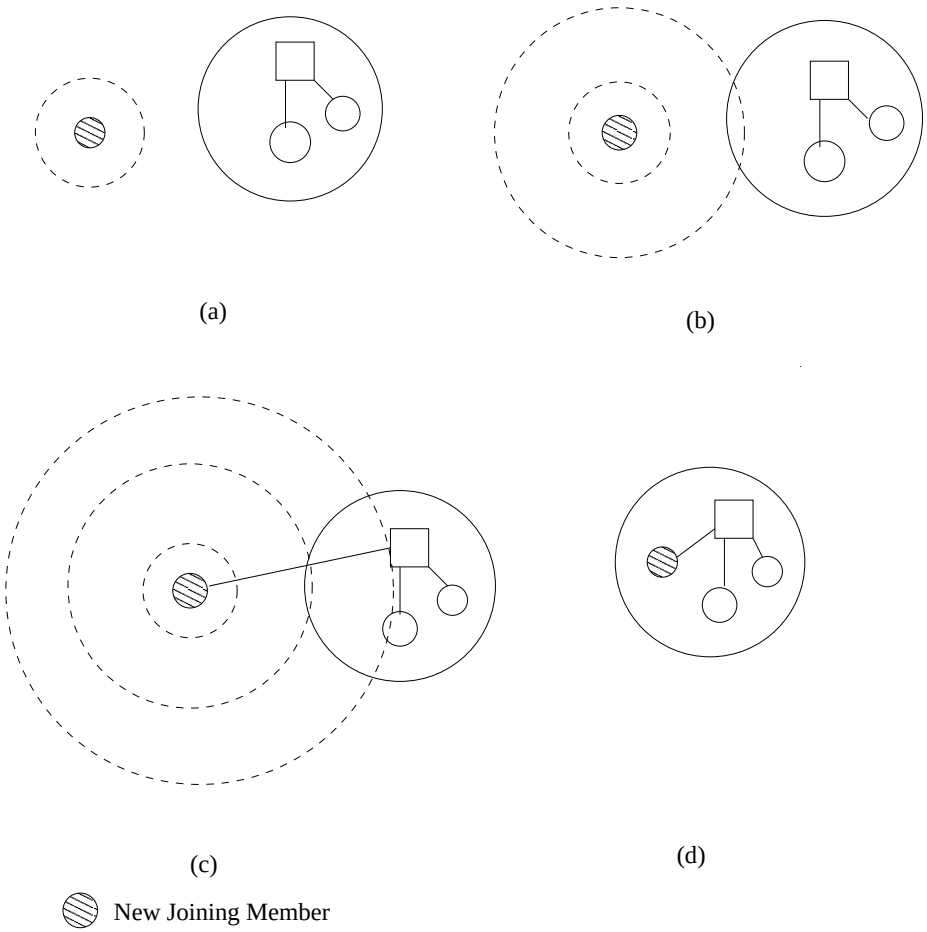

Figure 4.6: Expanding Ring Structure for Group Membership

Expanding ring structure is depicted in Figure 4.6 and works as follows. Any new member who wishes to join the multicast group sends out a GROUP_LEADER_SEARCH request message with a limited scope distance. The limited distance is achieved by setting a small TTL value in the request packet. If there is any Group Leader in the vicinity, then the Group Leader responds by sending a GROUP_LEADER_AVAILABLE response message. If the searching member does not receive any response from any Group Leader, it will start searching again, this time covering more distance. Increasing the TTL value in the

GROUP_LEADER_SEARCH packet again does this. On receipt of a response from a Group Leader, the new member may wish to join the group maintained by the Group Leader, or it may not wish to join the group. The number of members in the group may already be reaching the MAXIMUM_GROUP_MEMBERS limit or there may be a "better" group to join. This process will continue until a suitable Group Leader is found and the new member finds itself a place in a local group.

Another method by which a new member joins a local group is by the advertisement method as shown in Figure 4.7.

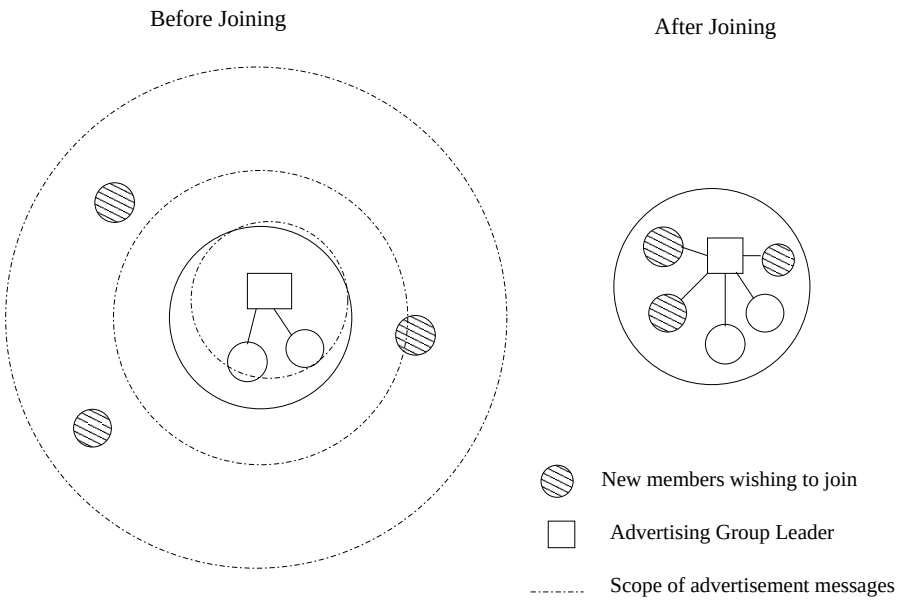

Figure 4.7: Group Leader's Advertisement for Membership

In the advertisement method, the Group Leaders send out a JOIN_MY_GROUP advertisement message periodically to the group-specific multicast address. Nodes who wish to join the multicast group listen for advertisement messages. The JOIN_MY_GROUP message contains information such as the distance of the Group Leader from the multicast source, the number of members already present in the group, data rate, delay, bandwidth, throughput and its error probability. The new member then calculates its distance from the Group Leader using the information in the Group Leaders advertisement message. It may wish to join the group or not

according to the parameters specified in the JOIN_MY_GROUP message. As all the Group Leaders send the JOIN_MY_GROUP message, a node is able to learn of all the Group Leaders present nearby and to gain partial information of the multicast tree. The time interval in which the JOIN_MY_GROUP message is sent should be decided in such a way that the control message traffic generated by the Group Leaders does not itself lead to congestion.

In both the methods, the new member that wishes to join the group sends an INTERESTED_IN_JOINING message to the Group Leader. The Group Leader responds positively with an ACK_JOIN message if it could accommodate the new member. On the contrary, the Group Leader sends a NACK_JOIN message to the requesting member indicating its inability to accommodate it.

In both the methods, if a new joining member cannot find a suitable Group Leader, then it may announce itself as a new Group Leader, leading to the formation of a new group. The new Group Leader could either attach itself directly to the source or to another Group Leader. It should be noted that, the advertisement method causes more control traffic than the expanded ring search method. Whenever the load of the network is low, the advertisement method will be adopted and when the load increases, group formation switches to the expanded ring search method.

4.6.2 Dynamic Reconfiguration of Groups

The local groups are able to re-organize themselves with the receivers changing to different groups according to the current network or congestion conditions. This is one of the most powerful features of protocol proposed in this thesis. Dynamic reconfiguration could be achieved in different ways, with the receivers shifting groups, Group Leaders shifting groups and the termination of receivers and Group Leaders. The following sections discuss all these aspects in detail.

Re-affiliation of members. The members of the multicast groups can re-arrange themselves by changing their membership to other local groups. This could happen for reasons such as the current group performing poorly, termination of Group Leader or the whole group. The re-affiliation could be categorized as follows.

Receivers shifting local groups. Receivers could shift to a different group as follows. Every Group Leader would send out a special control message called GL_STAT_MSG to every adjacent local domain group. The extent to which this control message is received is limited because the farther receivers would not be practically willing to change their groups at a long distance. Hence this message is sent such that only the adjacent group members receive it. This is taken care of by limiting the TTL field of the control message.

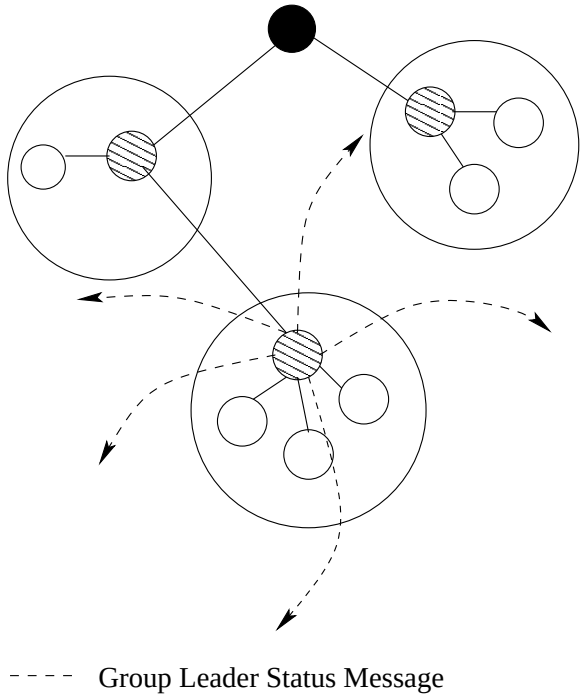

Figure 4.8: Group Leader's Status Message

All the members of the multicast group, viz., source, Group Leaders and the individual receivers, receive the control message sent by the Group Leader. The message contains most of the information similar to the Group Leader's advertisement message pertaining to a particular group's characteristics. These

include information on its distance from the source, data rate, delay, bandwidth, throughput, error probability, number of receivers in the group, number of packets actively held in the buffer, which would be a very good indicator of the congestion state of a particular group. On receipt of the GL_STAT_MSG, the individual receivers may shift to a different group because of the better service offered by the advertising Group Leader. Sending an INTERESTED_IN_JOINING to the Group Leader does this. The Group Leader responds positively with an ACK_JOINING message or negatively with a NACK_JOINING message. A receiver that wishes to change groups directly contacts the concerned Group Leader and to get its consent to join the particular group as a new member does.

Since the GL_STAT_MSG is very similar to the Group Leaders advertisement message, the Group Leader does not send out the status message when the group formation technique is through advertisement. Hence when the load of the network is low, the advertisement message acts as a status message too. This reduces the control traffic flow within the multicast group.

The drawback of this method is that, all the receivers that are in a bad state will try to move into the healthy group. The Group Leader would maintain a threshold in such a way that it does not permit over subscription membership more than the threshold. The allocation of the membership is on a first-come first-served basis.

Shifting of group leaders. It is possible that two Group Leaders could swap their local group leadership. This would happen because of the processing capability and memory of the Group Leader not sufficient enough to support its current members. Since the GL_STAT_MSG is received by the Group Leaders too, they could also contact their peers and initiate the swapping process, if desired. Sending an INTERESTED_IN_JOINING to the Group Leader does this. The Group Leader responds positively with an ACK_JOINING message or negatively with a NACK_JOINING message. It is also possible that the Group Leader of a particular

local group could swap its leadership with a local receiver of that group itself because of the reasons mentioned earlier.

Termination of group leader. A Group Leader might wish to terminate its operation at some point. This could happen when it wishes to leave the multicast group. Because of the termination of the Group Leader, some other node must be elected as the new Group Leader. Thus the Group Leader election process is invoked. There are two ways in which the Group Leader could be elected. The easiest method is for the old Group Leader to select a receiver in the local group to become the new Group Leader. The potential receiver should have the sufficient processing capability and memory to do the same. In another method, a newly joining member of the multicast group could also take over the group leadership if the above conditions are satisfied. In both the cases, the parent of the Group Leader is informed about the newly elected Group Leader. For security issues such as key management, the source should always have knowledge about the Group Leaders. Under such conditions, the termination or election of the Group Leaders are informed to the source.

Termination of local group. All or a large number of receivers could move to a different group because of the congestion state of the group. This shifting of the receivers could happen due to the GL_STAT_MSG exchange as mentioned earlier. If a majority of the receivers shift to different groups, the current membership of a group could become very thin. In such a case, the Group Leader would shed all the remaining receivers and terminate itself. When the Group Leader detects that the GROUP_MEMBER count is less than a threshold, it informs all the members about the termination of the group. The Group Leader receives a TERMINATE_ACK from all the members before terminating itself. This leads to the shutting down of the whole group.

Monitoring of unresponsive members and group leaders. The group members and Group Leaders could become unresponsive sometimes. The Group Leader buffers the data sent by the source until it gets an acknowledgement from each of its children. If a receiver becomes unresponsive, then the buffer begins to fill up. Therefore the receivers and the Group Leaders have to be monitored regularly. This is done using the GL_ALIVE control message sent by the Group Leaders. The mechanism is described in detail in Section 4.4. If a member of the group was found to be unresponsive by the Group Leader, the member is pruned from the group and no further repairs are served for the same. If the Group Leader was found to be unresponsive, the receiver would join to a different group.

Tree construction techniques. The construction of the hierarchical multicast tree has to be carefully managed for different types of multicast applications. Some multicast applications like video or teleconferencing consist of small number of receivers. Applications like stock quotes and content delivery potentially have a large number of receivers. Hence tree construction should be carefully managed for efficient multicast operation.

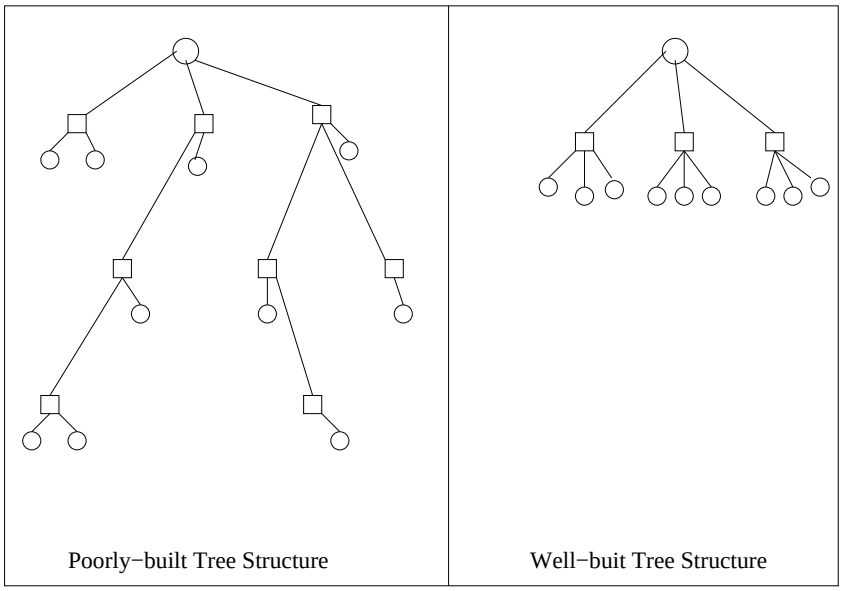

Figure 4.9: Tree Construction for an Application with Less Number of Receivers

Tree construction for small number of receivers. If the application consists of a small number of receivers, then the multicast tree construction is as follows. As the number of receivers is small, the Group Leader allows more receivers to join its group than its regular count. This will avoid the presence of a large number of Group Leaders each with a minimum number of receivers. The multicast operation would be better with limited number of groups when the number of receivers in the application is less. If the multicast tree consists of a number of Group Leaders with just a few members deep down, the end-to-end delay involved the transmission is increased as depicted in Figure 4.9. Increasing the MAXIMUM_GROUP_MEMBERS count would do this. Hence there would be limited number of groups with large population than large number of groups with sparse population. Hence the newly joining members would not be allowed to declare themselves as Group Leader unless the MAXIMUM_GROUP_MEMBERS threshold is exceeded.

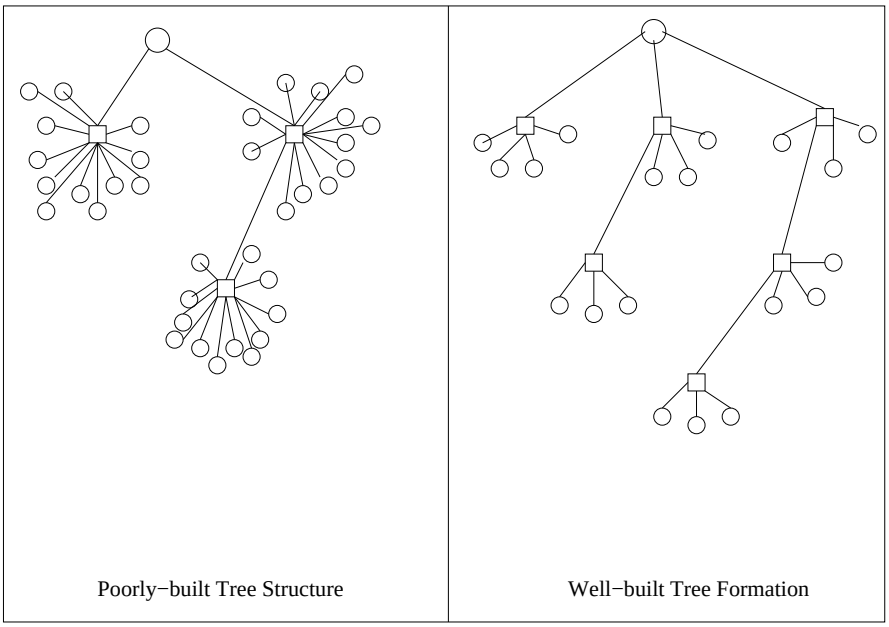

Figure 4.10: Tree Construction for an Application with Large Number of Receivers

Tree construction for large number of receivers. If the application involves large number of receivers, e.g., stock quotes and content delivery, then the multicast tree

construction would be as follows. The number of receivers in a local group would not be as large as the count for an application with small number of receivers. The MAXIMUM_GROUP_MEMBERS would be less when compared to the application with small number of receivers. Figure 4.10 depicts the poorly built tree with a local group serving a large number of receivers. The presence of large number of receivers would require the Group Leader to have sufficient processing power and memory. The multicast operation would be optimal if there is more number of local groups with less number of receivers to suit these types of applications. On the contrary if there were large number of receivers with few Group Leaders, it may lead to the ACK implosion problem at the Group Leader and may also result in congestion at the local groups.

In both the cases discussed above, there are other important factors that affect the proper tree construction. These include:

- *Distance or Hops*: The receivers of the multicast that reside in a network are grouped together to form a local group rather than grouping receivers that are in different networks wide apart. The number of hops or the distance of the receiver to its Group Leader also plays a vital role in correct formation of the groupings.

- *Maximum Branching Factor*: The maximum branching factor of the local group is proportional to a number of functions. The local processing power of the Group Leader is one factor that affects the branching factor. The Group Leader should have the sufficient processing power and capacity to accommodate the members of the local group. The multicast rate such as the amount of packets sent per second or the amount of bytes per packet also decides the branching factor of the group. The error rate and the load of the network is also a factor that influences the same.

4.7 Security Issues

The IETF has posted some evaluation criteria for reliable multicast transport protocols in [23]. Apart from the issues discussed above, it requires that a reliable multicast protocol should discuss the security issues.

The main objectives of multicast security is to preserve the **authentication** and **secrecy** of multicast data so that only the legitimate senders can send the data and only legitimate receivers can receive the data [23]. The secrecy of the multicast data is provided by public key cryptography mechanism. The data are encrypted using a group key, which is distributed among the receivers. This requires a good **group key management** solution in the protocol. To achieve this goal, the architecture of the protocol should be very well designed and should aid in the same.

The architecture defined in Section 3.2 satisfies both criteria of scalability and de-centralization. Due to the formation of local groupings, the protocol is scalable to a large receiver base. Also, the Group Leaders will act as a local key management entity managing a set of receivers, rather than a centralized controller. Hence if any of the local key controlling entities is down it could be taken over by another one.

The issues concerning the Group Key Management and access control are not considered in detail in this thesis.

4.8 Summary

The chapter provides a framework for reliable multicast protocol. The framework begins with the discussion of goals of the proposed protocol. The architectural design of the protocol is discussed followed by the mechanism of multicast data transfer. The section on acknowledgement and error recovery discusses the various types of retransmission performed by the Group Leader and source. The section also discusses the method by which the unresponsive receivers and Group Leaders are monitored and how the late joining receivers recover their data completely through central logging server. The section on congestion control discusses the control algorithms like slow start and congestion avoidance in detail. The group management section discusses the two group formation techniques of expanding

ring structure and advertisement method. Various other group management techniques like the dynamic reconfiguration of the receivers among different groups, group termination are discussed. The security issues are not discussed in detail.

CHAPTER 5
FUNCTIONAL MODEL

This chapter provides a functional model for the proposed reliable multicast protocol framework. The functional model identifies the various protocol components and its entities. The model also provides use case and sequence diagrams for basic operations of the protocol like multicast data transfer, acknowledgement, error recovery and group management. The use case and sequence diagrams are constructed based on the Unified Modeling Language (UML) specification. The functional model also provides an overall flow diagram of the protocol.

5.1 Protocol Components

There are four major components in the protocol design: Sender, Receiver, Group Leader, and Central Logging Server. The various components and their subcomponents are shown in Figure 5.1. The functions of each of them are detailed as follows.

5.1.1 Sender Component

The Sender component is responsible for the transmission of the multicast data to the whole group. In addition, it is also responsible for number of other functions like error recovery and congestion control, and it has the interface to interact with the Sender application. It has a number of subcomponents, details of which are listed below.

- *SNDR_TRANS_CONTROLLER*: This subcomponent has a number of modules that take care of the transmission of different types of packets.
- *SNDR_TR*: This is a module of SNDR_TRANS_CONTROLLER and is responsible for the transmission of new data packets.

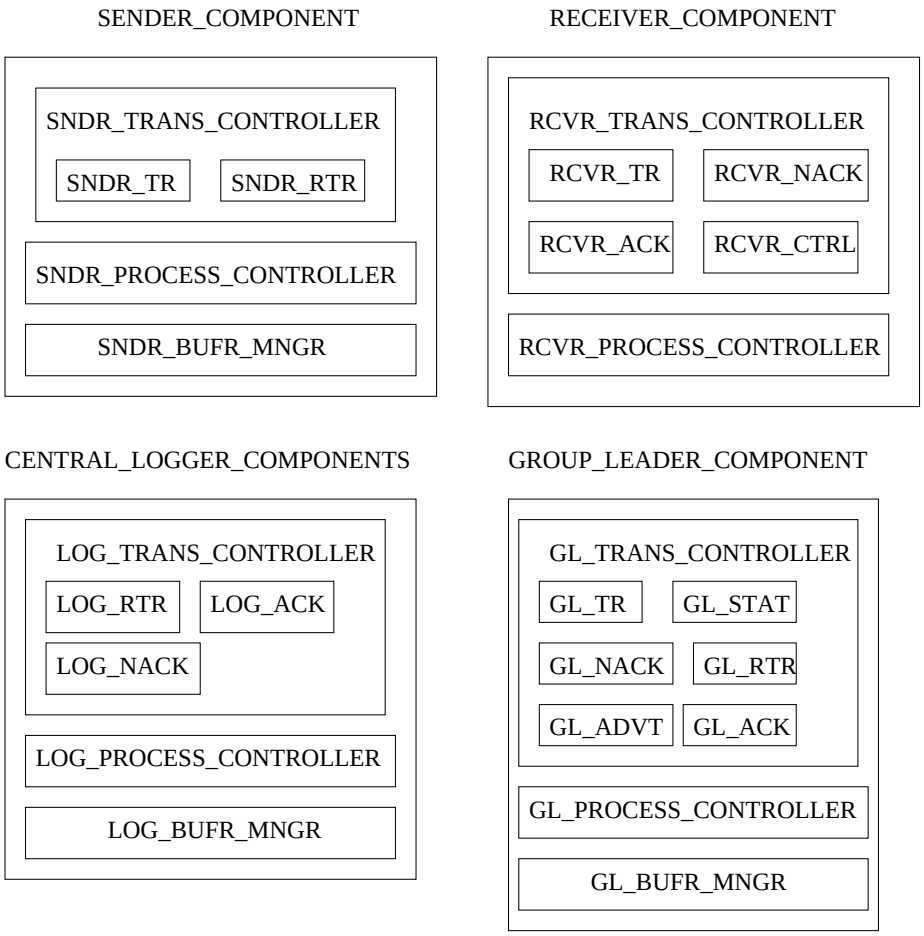

Figure 5.1: Functional Components of the Protocol

- *SNDR_RTR*: This is a module of SNDR_TRANS_CONTROLLER and is responsible for the retransmission of lost packets.

- *SNDR_PROCESS_CONTROLLER*: This subcomponent is responsible for the processing of ACK and NACK messages from Group Leaders and the Central Logging Server. It is also responsible for the processing of the control messages for group membership and congestion control.

- *SNDR_BUFR_MNGR*: This subcomponent is responsible for buffer management. The responsibilities include the buffering of messages and deleting them as they receive the ACKs from its children.

5.1.2 Receiver Component

The Receiver component delivers the received data packets to the receiver application. It also sends the ACK and NACK to the Group Leaders in accordance

with the reception of data packets. The various subcomponents of it are mentioned below.

- *RCVR_TRANS_CONTROLLER*: This subcomponent has a number of modules that take care of the transmission of different types of packets.

- *RCVR_TR*: This a module of RCVR_TRANS_CONTROLLER and is responsible for the transmission of the received data packets to the receiver application.

- *RCVR_ACK*: This a module of RCVR_TRANS_CONTROLLER and is responsible for the transmission of the ACK messages to the Group Leader.

- *RCVR_NACK*: This a module of RCVR_TRANS_CONTROLLER and is responsible for the transmission of the NACK messages to the Group Leader and the Central Logging Server to request retransmission of lost packets.

- *RCVR_CTRL*: This a module of RCVR_TRANS_CONTROLLER and is responsible for the transmission of the control packets for group management techniques and congestion control.

- *RCVR_PROCESS_CONTROLLER*: This subcomponent is responsible for processing of the control messages for group membership and congestion control.

5.1.3 Group Leader Component

The Group Leader component is responsible for buffering data packets from the sender temporarily. It is also involved in the local error recovery with the receivers by the retransmission of lost packets. The Group Leader is also responsible for detecting the congestion and reporting the same up the multicast tree. The various subcomponents of the Group Leader are mentioned below.

- *GL_TRANS_CONTROLLER*: This subcomponent has a number of modules that take care of the transmission of different types of packets.

- *GL_TR*: This a module of GL_TRANS_CONTROLLER and is responsible for the transmission of the received data packets to the receiver application.

- *GL_ACK*: This a module of GL_TRANS_CONTROLLER and is responsible for the transmission of the ACK messages to the source or other Group Leader up in the hierarchy.

- *GL_RTR*: This a module of GL_TRANS_CONTROLLER and is responsible for the retransmission of the lost packets.

- *GL_NACK*: This a module of GL_TRANS_CONTROLLER and is responsible for the transmission of the NACK messages to another Group Leader up in the hierarchy or to the Central Logging Server to request retransmission of lost packets.

- *GL_ADVT*: This a module of GL_TRANS_CONTROLLER and is responsible for the transmission of the group advertisement packets.

- *GL_STAT*: This a module of GL_TRANS_CONTROLLER and is responsible for the transmission of the local group status messages to the multicast group.

- *GL_PROCESS_CONTROLLER*: This subcomponent is responsible for the processing of ACK and NACK messages received from the receivers and child Group Leaders. It is also responsible for processing of the control messages like group membership and congestion control.

- *GL_BUFR_MNGR*: This subcomponent is responsible for buffer management. The responsibilities include the buffering of messages and deleting them as they receive the ACKs from its children.

5.1.4 Central Logging Server Component

The central logger component is responsible for the complete data recovery. It buffers the data permanently in the disk and recovers the receivers and Group Leaders in case of loss. The various subcomponents of it are discussed as follows.

- *LOG_TRANS_CONTROLLER*: This subcomponent has a number of modules that takes care of the transmission of different types of packets.

- *LOG_RTR*: This is a module of LOG_TRANS_CONTROLLER and is responsible for the retransmission of lost packets.

- *LOG_ACK*: This is a module of LOG_TRANS_CONTROLLER and is responsible for the transmission of ACK packets to the sender.

- *LOG_NACK*: This is a module of LOG_TRANS_CONTROLLER and is responsible for the transmission of the NACK packets to the source.

- *SNDR_PROCESS_CONTROLLER*: This subcomponent is responsible for the processing of NACK messages from Group Leaders and the receivers.

- *LOG_BUFR_MNGR*: This subcomponent is responsible for buffer management. The responsibilities include the buffering of messages in disks and retrieving them during retransmissions.

Message	Description
GL_SRCH	Message sent by clients in search of Group Leaders
INT_IN_JOIN	Message sent by clients in response to the join reply from Group Leaders
GRP_JOIN_ACK	Acknowledgement message sent by Group Leader for group membership
JOIN_MY_GRP	Advertisement message sent by Group Leader for group membership
GL_STATUS	Status message sent by Group Leader to aid the group member's re−affiliation
LEAVE_GRP	Message sent by receivers informing the Group Leader before leaving the group
LEAVE_GRP_ACK	Acknowledgement for the previous message
FIND_GRP_LDR	Message sent by Group Leader to the local group to initiate leader election process
TERM_ACK	Message sent by receivers acknowledging the termination of the Group Leader
GL_ALIVE	Message sent by Group Leaders, which aid the receivers in finding the unresponsive Group Leaders

Figure 5.2: Different Messages Used in Protocol Operation

Appendix A contains pseudo algorithms for the various components of the protocol. Though the operations performed by them are listed sequentially, they are performed concurrently. Appendix B contains the pseudo algorithms for various group management techniques between the Group Leaders and receivers. The various messages involved in the protocol operation are listed in Figure 5.2.

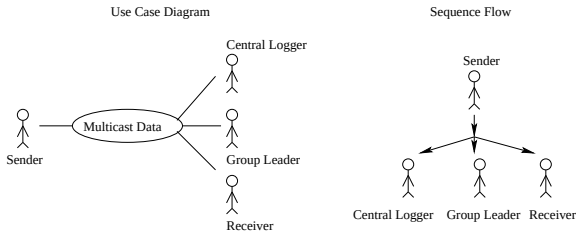

Figure 5.3: Use Case and Sequence Analysis for Multicast Data Operation

5.2 Use Case and Sequence Diagrams

This section deals with the use case design for the various phases of the protocol operation. The use case analysis depicts the various operations phases of the

protocol as different cases and analyzes them with the major players involved with it in an abstract way without any implementation details.

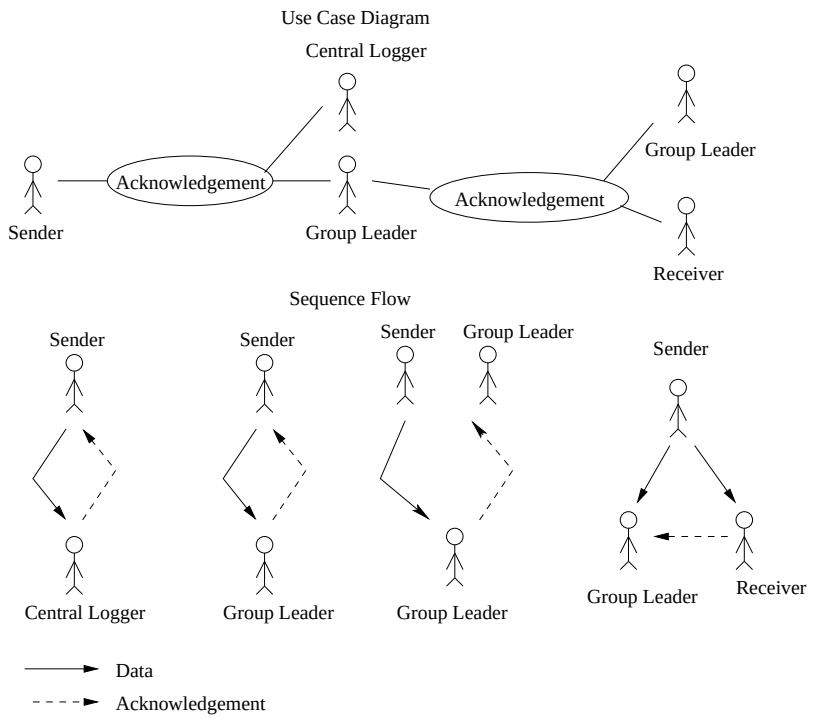

Figure 5.4: Use Case and Sequence Analysis for Acknowledgement

Multicast data analysis. Figure 5.3 depicts the use case diagram and the sequence flow associated with the operation of multicast data transfer. The sender multicasts the data packets to the whole group and they are received by all the components of the group. The direction of data flow from sender to the components, central log server, Group Leaders and receivers are shown as sequence flows in Figure 5.3.

Acknowledgement analysis. Figure 5.4 depicts the use case diagram and the sequence flow associated with the acknowledgement packets. The data packets sent by the sender are acknowledged by the Group Leaders attached directly to it and the central log server. Group Leaders are acknowledged by the receivers, and the Group Leaders attached below it. There are four different sequence diagrams involved with the second part of Figure 5.4. Both the central log server and the

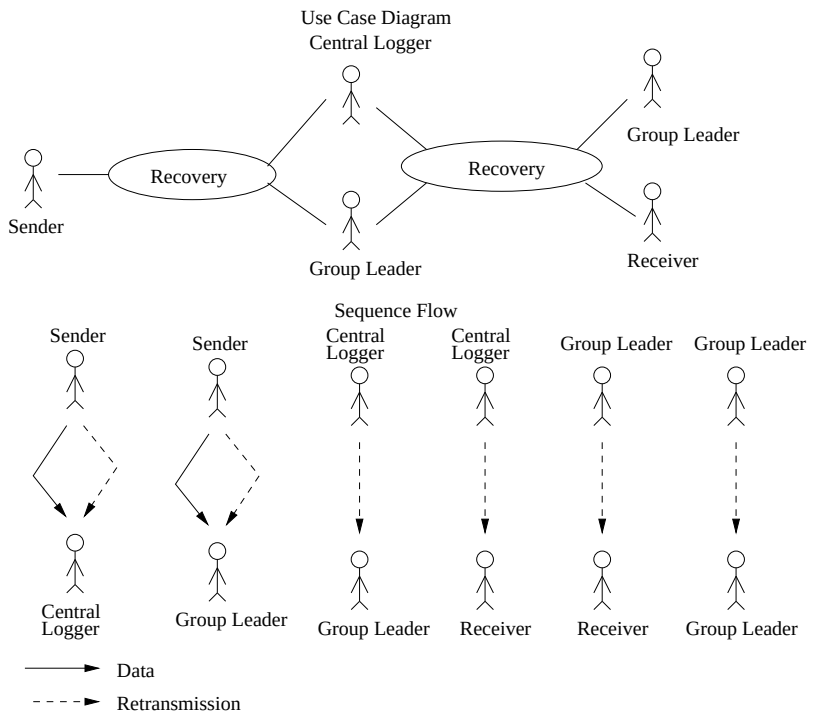

Figure 5.5: Use Case and Sequence Analysis for Error Recovery

Group Leaders that are attached directly to the source, send their acknowledgement to sender in response to the data packets received from it. The receivers send their acknowledgements to the Group Leader rather than the sender and Group Leaders acknowledge their parent Group Leaders. The different levels in the multicast tree improve the scalability of the protocol. The flow in all the four cases are represented as sequence flows in Figure 5.4.

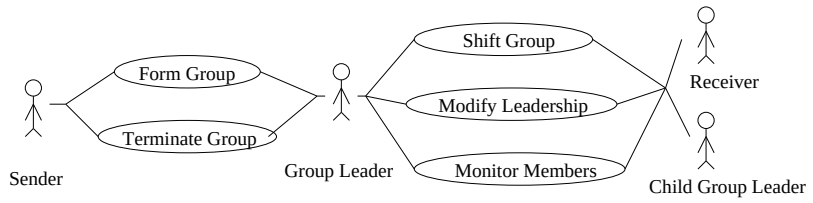

Figure 5.6: Use Case and Sequence Analysis for Group Management

Error recovery analysis. Figure 5.5 depicts the use case diagram and the sequence flow associated with the error recovery packets. Error recovery is required in response to the packets lost by the various components of the multicast group. There are six different sequences associated with the error recovery. The sender

retransmits lost packets to the central log server and its children Group Leaders. The central log server retransmits the packets lost both by the receivers and the Group Leaders. The Group Leaders also retransmits the packets for the receivers and their Group Leader children. The flow in all the six cases is represented as sequence flows in Figure 5.5.

Group management analysis. The use case diagram for the Group Management is depicted in Figure 5.6. The sender and the Group Leaders are involved in activities such as formation and termination of the groups. The other operations such as shifting among the group members, modifying the leadership and monitoring of the members are restricted to the receivers and the Group Leaders.

Overall sequence analysis. The Figure 5.7 represents the combined sequence diagram of all the operations represented above. This sequence diagram represents the order in which the protocol operation takes place. The four entities are represented along with an extra Group Leader and an extra receiver to show the retransmission operation involved. Each arrow represents the flow of packets from a source entity to a destination entity in the direction of the pointing arrowhead. They are marked as X.Y with X being the sending entity and Y being the sequence number of the sending entity. It is seen that the source multicasts the data packet to all the entities and everyone acknowledges it promptly. Receiver R1 acknowledges the data packet to its Group Leader whereas R2 requests a retransmission. After the reception of the retransmission, R2 acknowledges the same. The other details pertaining to the retransmissions by central log server and source, congestion control schemes and the group management are not shown in Figure 5.7.

5.3 Flow Diagram

Figure 5.8 depicts data flow and control flow inside a component of the protocol. The component could be a sender or a Group Leader or the central log server. The

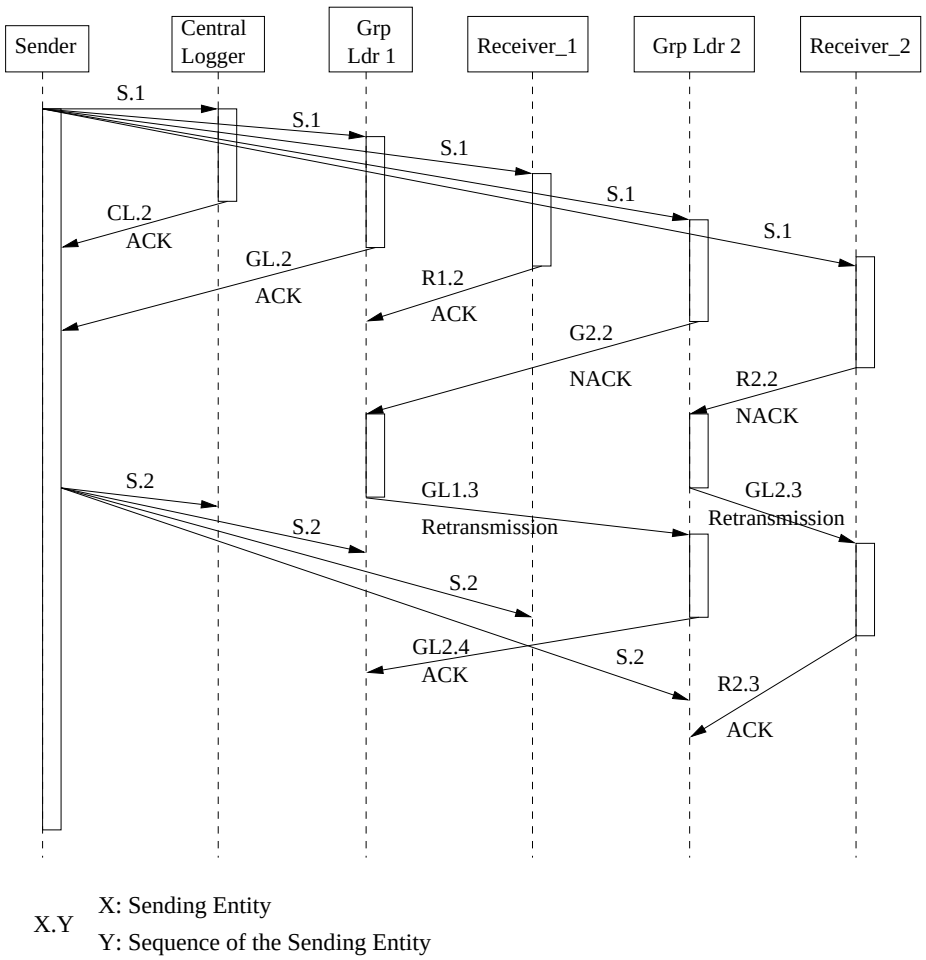

Figure 5.7: Sequence Diagram of the Protocol

component depicted in Figure 5.8 could not be a receiver as there are buffer mechanisms involved. The data flow is depicted in regular dark line while the control flow is depicted in dashed line. There are two separate buffers for transmission and retransmission. Similarly there are separate transmission mechanisms for multicast and unicast. There are separate entities for managing the control messages for group management and flow control. An important feature of the flow inside the component is that the data flows in from the top and flows out through the bottom. Also, the control flows in from the sides and flows out through the sides. Separate queues are provided for ACK and NACK.

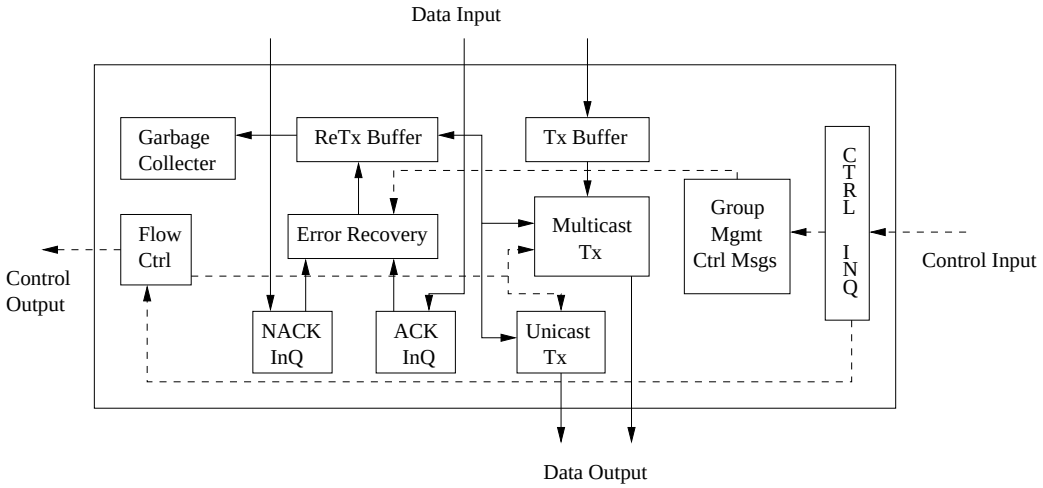

Figure 5.8: Flow Diagram of a Component

5.4 Summary

This chapter provided a functional model of the protocol framework. The functional model is intended to provide an abstract overview of the protocol without concentrating on the implementation details. This model could also serve as an interface between the functional people and technical team. The model has use case analysis, sequence analysis, flow diagrams, and sequence diagrams. These were drawn according to the UML specifications.

CHAPTER 6
CONCLUSION AND FUTURE WORK

6.1 Conclusion

Reliable multicast refers to the reliable manner in which a message is sent from a sender to a set of receivers. It is still an active area of research with several works in literature. Most of the works are classified as sender-initiated and receiver-initiated approaches. Although there are a number of protocols in existence, most of them were developed with a particular application in mind. There are several important issues that make the design of a reliable multicast protocol difficult. The protocol has to be scalable to a large number of receivers. Since several receivers are involved, the implosion problem must be addressed. Implosion at the sender occurs when all the receivers send their response traffic back to source. The protocol should provide recovery mechanisms that minimize the control traffic and recovery time. The protocol must also provide mechanisms to recover from congestion in the network. Management mechanisms of different entities in the protocol must be mentioned. All these factors make design of the reliable multicast protocol challenging.

Some of the popular reliable multicast protocols in existence are LGMP, RMTP, TRAM, SRM, LBRM, and XTP. An analysis of these protocols with respect to the issues mentioned previously is mentioned in Chapter 3. The concept of logically grouping the receivers was proposed in LGC. Although LGC recovered the packets locally, it had to go to the sender if none of the members have the missing packet. LGC did not mention congestion control. The local grouping concept proposed by LGC was later adopted in the designs of RMTP and TRAM. RMTP builds a number of local subtrees, which together form the global multicast tree. Although

RMTP is scalable to a large number of receivers, it does not support much in the way of group management techniques. RMTP does not provide end-to-end congestion control with any feedback sent from the receivers. Multicast in TRAM is based on the repair tree construction. TRAM has several features like local grouping, tree construction based upon LGC and RMTP. TRAM provides algorithms for optimized tree construction techniques suitable for a variety of multicast applications. While, TRAM does not support complete data recovery, it does mentions a number of control and status messages for performing the group management techniques. LBRM proposes the idea of distributed logging of data packets, which aids in local recovery and reduces the end-to-end propagation delay. Although the receivers are grouped together forming sites, LBRM does not specify group management techniques. LBRM was developed for high performance simulation applications that require low-latency packet loss detection. Hence the protocol has the ability to provide keep-alive or heartbeat data packets, even when the application does not provide it. SRM was developed for supporting a distributed whiteboard application. There is no local grouping of receivers, which severely limits the scalability of the protocol. Retransmission requests made by the receivers are multicast to the whole group, and other receivers that require the same packets back off looking the request. MTCP has a hierarchical structure of multicast receivers with source rooted on top of the tree. It does not logically group the receivers, which again limits its scalability of the receivers. It proposes a congestion control algorithm based on hierarchical feedback sent by the receivers up towards the source.

Contribution. This thesis proposes a framework for a reliable multicast protocol that is scalable to a large number of receivers and is rich in group management techniques. The important features of the protocol are discussed as follows.

- The architectural design of the protocol with the local groupings of receivers arranged hierarchically makes it scalable to a large number of receivers.
- Local recovery of packets at the Group Leaders reduces the round trip time involved in retransmission.
- Reporting of ACKs and NACKs by the receivers to the Group Leaders eliminates the ACK implosion problem by design.
- Complete recovery of data packets for the receivers joining late in the multicast is accomplished through the central logging server.
- Unresponsive receivers and Group Leaders are monitored by a special control mechanism.
- The protocol provides a receiver-based congestion control mechanism along with TCP-like slow start congestion control algorithms at the source that makes the protocol TCP-friendly.
- Receivers and Group Leaders report the congestion notifications in a hierarchical fashion to the source.
- The protocol provides two different group formations techniques: expanding ring and advertisement method that are adapted dynamically to suit the network load.
- Receivers can dynamically reconfigure themselves by changing the group membership with the help of status messages from Group Leaders.

The framework also includes a list of pseudo algorithms for the multicast operation and group management techniques. A functional model detailing the various components of the protocol was proposed. The model also included a list of use-case and sequence diagrams for the basic multicast operation of the protocol.

Some of the features in the protocol framework were adapted from the earlier works. The concepts of local groping of receivers were adopted from LGC, but the features have been modified to suit the current framework. Several new features like the dynamic group management techniques and complete data recovery through central logger were incorporated.

6.2 Future Work

The following are the several areas in which the future work could be done.

- The framework discussed in thesis addresses only the point-to-multipoint communication. For multipoint-to-multipoint communications with several senders, multicast trees would have to be setup at each sender.

- The framework does not discuss the security aspects in detail such as source authentication, access control and the key management techniques.

- Tree optimization techniques are not discussed in the framework, although a mention of factors that affect the correct tree-formation are specified. Dynamic optimization by continually reconfiguring the tree structure by breaking or branching the tree to suit the network conditions is an interesting area to address.

- The presence of central log server helps in the complete data recovery. Multiple permanent log servers could be distributed to reduce the propagation delay in retransmission and for scaling purposes. The tradeoff involved in the introduction of distributed log servers versus the buffering of data at the Group Leaders could be studied.

The protocol proposed in this provides a framework along with a functional model for a reliable multicast protocol. It also provides pseudo algorithms for the multicast operation and group management techniques. The proposed protocol framework is for a point-to-multipoint communications. Analysis and issues relating to the multipoint-to-multipoint communications could be studied further.

APPENDIX A
PSEUDO ALGORITHMS FOR MULTICAST OPERATION

Algorithm 1: Sender Operation

```
while Connection is Open do
    Receive Data from Sending Application;
    Multicast Data packets to the Group;
    Buffer the packets before sending;
    Process Acknowledgements and Retransmissions;
    Manage buffered Data packets;
    if Notified for Congestion then
        Reduce the sending rate;
    end
    Perform Group Management activities;
end
```

Algorithm 2: Central Log Server Operation

```
while True do
    Receive Data packets from Sender;
    Send ACK or NACK to Sender;
    if Request for Recovery then
        Perform Retransmissions;
    end
    Store and Retrieve Data from Disk;
end
```

Algorithm 3: Group Leader Operation

```
while True do
    Receive Data packets from Sender;
    Buffer Data packets;
    Send ACK or NACK to Parent;
    Process ACK or NACK from Children;
    Manage the buffered Data;
    if Request for Recovery then
        Perform Retransmissions;
    end
    Perform Group Management activities;
    if Congestion or problem then
        Send Notification to Parent;
    end
    Send received packets to Receiving Application;
end
```

Algorithm 4: Receiver Operation

```
while True do
    Receive Data packets from Sender;
    Send ACK or NACK to Group Leader;
    Perform Group Management Activities;
    if Congestion or problem then
        Send Notification to Group Leader;
    end
    Send received packets to Receiving Application;
end
```

APPENDIX B
PSEUDO ALGORITHMS FOR GROUP MANAGEMENT

Algorithm 5: Expanding Ring Search: Client

```
while GroupLeaderFound == False do
    Multicast a GROUP_LEADER_SEARCH message;
    Collect Responses;
    if No Replies then
        Multicast the GROUP_LEADER_SEARCH message with longer
        TTL;
    else
        foreach Reply collected from GroupLeader do
            Analyze the reply for best characteristics;
        end
        Send an INTERESTED_IN_JOINING message to GroupLeader;
        if ACK_JOINING received then
            GroupLeaderFound ← True;
        else
            Continue Searching;
        end
    end
end
```

Algorithm 6: Expanding Ring Search: GroupLeader

```
while True do
    Collect messages from all potential receivers;
    if GROUP_LEADER_SEARCH message then
        foreach Reply collected do
            if MAX_MEMBERS not exceeded then
                Send Reply with the Group characteristics;
            end
        end
    else if INTERESTED_IN_JOINING message then
        Send ACK_JOINING message;
end
```

Algorithm 7: Advertisement Method: Client

```
while GroupLeaderFound == True do
    Collect the JOIN_MY_GROUP advertisement from GroupLeader;
    foreach Advertisement message collected do
        Analyze and Categorize for best characteristics;
        Choose a Group Leader reachable with minimum hops;
        if Group Leaders MAX_MEMBERS not exceeded then
            Send INTERESTED_IN_JOINING message;
            Wait for ACK_JOINING from GroupLeader;
            if ACK_JOINING received is Success then
                GroupLeaderFound ← True;
            else
                Continue Searching;
            end
        else
            Analyze another Reply;
        end
    end
end
```

Algorithm 8: Advertisement Method: Group Leader

```
while True do
    Send JOIN_MY_GROUP at certain Time Intervals;
    Collect responses from receivers;
    if INTERESTED_IN_JOINING message then
        if MAX_MEMBERS not exceeded then
            Send Positive ACK_JOINING message;
            Increment GROUP_MEMBER count;
        else
            Send Negative ACK_JOINING message;
        end
    end
end
```

Algorithm 9: Receivers Shifting Local Group: Client

```
while True do
    Collect the GROUP_LDR_STATUS message from Group Leader(s);
    foreach Status message collected do
        Analyze and Choose the Group Leader with best characteristics;
        Send INTERESTED_IN_JOINING message;
        if Positive ACK_JOINING is received then
            Send LEAVE_GROUP to Group Leader;
            Wait for the receipt of LEAVE_GROUP_ACK;
            Join the new group;
        else
            Analyze another status message;
        end
    end
end
```

Algorithm 10: Receivers Shifting Local Group: Group Leader

```
while True do
    Send GROUP_LDR_STATUS message at regular Time Intervals;
    Collect responses from receivers;
    if INTERESTED_IN_JOINING message then
        if MAX_MEMBERS not exceeded then
            Send Positive ACK_JOINING message;
            Increment GROUP_MEMBER count;
        else
            Send Negative ACK_JOINING message;
        end
    end
end
```

Algorithm 11: Group Leader Shift or Termination: Group Leader

```
if GROUP_MEMBER count less than Threshold then
    Perform regular Group Leader activities;
    if wish to leave or terminate then
        Inform all receivers about termination;
        Receive TERMINATE_ACK from all receivers;
        Fulfill all pending obligations;
        Initiate the FIND_GROUP_LEADER operation;
        Terminate;
    end
else
    Inform all receivers about termination;
    Receive TERMINATE_ACK from all receivers;
    Inform source about termination of group;
    Shed all clients;
    Terminate receivers and self;
end
```

Algorithm 12: Monitoring of Unresponsive Group Leader: Receiver

```
while True do
    Receive the GL_ALIVE messages;
    if Do not receive GL_ALIVE messages then
        Mark its absence in the ACK message;
        Send the marked ACK to Group Leader;
        Wait for the response;
        Send 3 marked ACKs until any response;
        if No Response from Group Leader then
            Group Leader is dead;
            Initiate algorithm to join a new group;
        else
            Group Leader is Alive;
            Remain in the same group;
        end
    end
end
```

Algorithm 13: Monitoring of Unresponsive Receivers: Group Leader

```
while True do
    Send GL_ALIVE messages periodically;
    Receive the ACK from the receivers;
    if Do not receive 3 ACK messages continually then
        Unicast a GL_ALIVE to the receiver;
        Wait for the response;
        if No Response from the receiver then
            Receiver is dead or left the group;
            Terminate the receiver from the group;
        else
            Receiver is Alive;
            Continue Multicast Operation;
        end
    end
end
```

REFERENCES

[1] Information Sciences Institute, "Internet Protocol," Request for Comment (RFC) 791, Internet Engineering Task Force, September 1981

[2] Information Sciences Institute, "Transmission Control Protocol," Request for Comment (RFC) 793, Internet Engineering Task Force, September 1981

[3] S. Floyd, V. Jacobson, C. Liu, S. McCanne, L. Zhang, ''A Reliable Multicast Framework for Light-weight sessions and Application Level Framing," IEEE/ACM Transactions on Networking, vol. 5, no. 6, pages 784-803, December 1997

[4] J. W. Atwood, O. Catrina, J. Fenton, W. T. Strayer, "Reliable Multicasting in Xpress Transport Protocol," IEEE Conference on LCN, pages 202-211, Minneapolis, MN, USA, October 1996

[5] H. W. Holbrook, S. K. Singhal, D. R. Cheriton, "Log-based Receiver-reliable Multicast for Distributed Interactive Simulation," SIGCOMM, pages 328-341, 1995

[6] M. Hofmann "A Generic Concept for Large-Scale Multicast," International Zurich Seminar on Digital Communications, pages 95-106, 1996

[7] I. Rhee, N. Ballaguru, G. Rouskas, "MTCP: Scalable TCP-like Congestion Control for Reliable Multicast," IEEE INFOCOM, vol. 3, pages 1265-1273, New York, NY, USA, March 1999

[8] S. Paul, K. Sabnani, J. Lin, S. Bhattacharyya "Reliable Multicast Transport Protocol (RMTP)," IEEE Journal of Selected Areas in Communications, vol. 15, pages 407-421, April 1997

[9] D. Chiu, S. Hurst, M. Kadansky, "TRAM: A Tree-Based Reliable Multicast Protocol," Technical Report TR-98-66, Sun Microsystems, July 1998

[10] W. Stallings, *Cryptography and Network Security: Principles and Practice*, Second Edition, Prentice Hall, Upper Saddle River, New Jersey, July 1998

[11] V. Roca, L. Costa, R. Vida, A. Dracinschi, S. Fdida, "A Survey of Multicast Technologies," Technical Report, LNCS 2236, September 2000

[12] B. Quinn, K. Almeroth, "IP Multicast Applications: Challenges and Solutions," Request for Comment (RFC) 3170, Internet Engineering Task Force, September 2001

[13] B. N. Levine, J. J. Garcia-Luna-Aceves, "A Comparison of Reliable Multicast Protocols," International Conference on Network Protocols, pages 112-121, Columbus, OH, USA, October 1996

[14] K. Obraczka "Multicast Transport Protocols: A Survey and Taxonomy," IEEE Communications Magazine, vol. 36, pages 94-102, Toronto, Ont., Canada, January 1998

[15] J. Postel, Information Sciences Institute, "User Datagram Protocol," Request for Comment (RFC) 768, Internet Engineering Task Force, August 1980

[16] V. Jacobson "Congestion Avoidance and Control," ACM SIGCOMM, vol. 18, pages 314-329, Stanford, CA, USA, August 1998

[17] X. Xiao, L. M. Ni, "Internet QoS: A Big Picture," IEEE Network, vol. 13, pages 8-18, March-April 1999

[18] A. Mankin, A. Romanow, S. Bradner, V. Paxson, "IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols," Request for Comment (RFC) 2357, Internet Engineering Task Force, June 1998

[19] J. Nonnenmacher, M. Lacher, M. Jung, E. Biersack, G. Carle, "How bad is Reliable Multicast without Local Recovery ?," INFOCOM, vol. 3, pages 972-979, San Francisco, CA, USA, April 1998

[20] S. K. Kasera, J. Kurose, D. Towsley, "A Comparison of Server-based and Receiver-based Local Recovery Approaches for Scalable Reliable Multicast," Technical Report UM-CS-1997-069, 1997

[21] S. J. Golestani, K. Sabnani, "Fundamental Observations on Multicast Congestion Control in the Internet," INFOCOM, vol. 2, pages 990-1000, New York, NY, USA, March 1999

[22] S. Floyd, M. Handley, "Requirements for Congestion Control for Reliable Multicast," Reliable Multicast Workshop in Cannes, September 1997

[23] M. J. Moyer, J. R. Rao, P. Rohatgi, "A Survey of Security Issues in Multicast Communications," IEEE Network, vol. 13, pages 12-23, November-December 1999

BIOGRAPHICAL SKETCH

Venkata L. Ramasubramaniam was born in Srivilliputtur, Tamil Nadu, India, on October 17, 1978. He earned his high school diploma from Sir. M. Venkata Subba Rao matriculation school. He graduated with a Bachelor of Engineering degree with distinction in computer science and engineering in 1999 from Madurai Kamaraj University, Madurai. He came to the University of Florida (in Gainesville, Florida) to pursue a Master of Science degree in the Computer and Information Science and Engineering.