PERFORMANCE ANALYSIS OF RELIABLE MULTICAST PROTOCOLS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

COŞKUN ÇELİK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

DECEMBER 2004

Approval of the Graduate School of Natural and Applied Science

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof Dr. İsmet Erkmen
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Cüneyt F. Bazlamaççı
Supervisor

Examining Committee Members

Prof. Dr. Hasan Güran	(METU,EE)	
Asst. Prof. Dr. Cüneyt F. Bazlamaççı	(METU,EE)	
Prof. Dr. Semih Bilgen	(METU,EE)	
Dr. Ece Güran	(METU,EE)	
Dr. Altan Koçyiğit	(METU,IS)	

ABSTRACT

PERFORMANCE ANALYSIS OF RELIABLE MULTICAST PROTOCOLS

Çelik, Coşkun
M.Sc. Department of Electrical and Electronics Engineering
Supervisor: Asst. Prof. Dr. Cüneyt F. Bazlamaçcı

December 2004, 81 pages

IP multicasting is a method for transmitting the same information to multiple receivers over IP networks. Reliability issue of multicasting contains the challenges for detection and recovery of packet losses and ordered delivery of the entire data. In this work, existing reliable multicast protocols are classified into three main groups, namely tree based, NACK-only and router assisted, and a representative protocol for each group is selected to demonstrate the advantages and disadvantages of the corresponding approaches. The selected protocols are SRM, PGM and RMTP. Performance characteristics of these protocols are empirically evaluated by using simulation results. Network Simulator-2 (ns2), a discrete event simulator is used for the implementation and simulation of the selected protocols. The contributions of the thesis are twofold, i.e. the extension of the ns library with an open source implementation of RMTP which did not exist earlier and the evaluation of the selected protocols by investigating performance metrics like distribution delay and recovery latency with respect to varying multicast group size, network diameter, link loss rate, etc.

Keywords: IP Multicasting, Reliable Multicast, Network Simulation, Performance Analysis

ÖZ

GÜVENİLİR ÇOKLU YAYILIM PROTOKOLLERİNİN PERFORMANS ANALİZİ

Çelik, Coşkun
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Danışmanı : Yrd. Doç. Dr. Cüneyt F. Bazlamaçcı

Aralık 2004, 81 sayfa

IP çoklu yayılım, aynı bilginin İnternet Protokolü kullanan ağlar üzerinden bir çok alıcıya gönderilmesi için kullanılan bir yöntemdir. Çoklu yayılımda güvenilirlik sorunu paket kayıplarının belirlenmesi ve giderilmesini ve tüm verinin sıralı bir şekilde aktarılmasını içerir. Bu çalışmada mevcut çoklu yayılım protokolleri, ağaç yapılı, negatif bildirimli ve yönlendirici destekli olarak üç ana grupta sınıflandırılmıştır ve her grubun avantaj ve dezavantajlarını incelemek için bir protokol seçilmiştir. Seçilen protokoller SRM, PGM ve RMTP'dir. Bu protokollerin performans özellikleri benzetim sonuçları kullanılarak deneysel olarak değerlendirilmiştir. Protokollerin benzetimi için bir kesikli olay benzetim aracı olan Network Simulator-2 (ns2) kullanılmıştır. Tezin iki yönlü katkısı bulunmaktadır; bunlar daha önceden varolmayan bir RMTP açık kaynak kodu ile ns kütüphanesinin genişletilmesi ve seçilmiş protokollerin, yayılım zamanı, yenileme gecikmesi gibi performans metriklerinin değişen grup büyüklüğü, ağ genişliği, hatlardaki kayıp oranı vb. göre incelenmesi ve değerlendirilmesidir.

Anahtar Kelimeler: IP Çoklu Yayılım, Güvenilir Çoklu Yayılım, Ağ benzetimi, Performans analizi.

ACKNOWLEGMENTS

The author wishes to express his deepest gratitude to his supervisor Asst. Prof. Dr. Cüneyt F. Bazlamaçcı for his guidance, advice, criticism, encouragements and insight throughout the research.

He would like to express his appreciation to the members of Electrical and Electronics Engineering Department of Gazi University, who provided technical support and tolerance during this study.

The author would also like to thank his mother Nursen Çelik, his father Süleyman Çelik and his brother M. Orkun Çelik for their endless support and patience. And last, he wishes to thank his friends for their encouragement, love, and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ACK	Positive Acknowledgment
ALC	Asynchronous Layered Coding
AP	Acknowledgement Processor
ARP	Address Resolution Protocol
ARQ	Automatic Repeat Request
BGMP	Border Gateway Multicast Protocol
CBR	Constant Bit Rate
CBT	Core-Based Tree
DHCP	Dynamic Host Configuration Protocol
DR	Designated Receiver
DVMRP	Distance-Vector Multicast Routing Protocol
FEC	Forward Error Correction
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Number Authority
IGMP	Internet Group Management Protocol
IP	Internet Protocol
LBRM	Log-based Receiver-Reliable Multicast
LMS	Lightweight Multicast Service
MDP	Multicast Dessemination Protocol
MFTP	Multicast File Transfer Protocol
MOSPF	Multicast OSPF
MTP	Multicast Transport Protocol
NACK	Negative Acknowledgment
NCF	NACK Confirmation
NE	Network Element

NS-2	Network Simulator - 2
ODATA	Original Data
OSPF	Open Shortest Path First
PGM	Pragmatic General Multicast
PIM-DM	Protocol-Independent Multicast – Dense Mode
RAMP	Reliable Adaptive Multicast Protocol
RBP	Reliable Broadcast Protocol
RDATA	Repair Data
RFC	Requests For Comments
RMCM	Reliable Multicast For Core-Based Multicast Trees
RMP	Reliable Multicast Protocol
RMTP	Reliable Multicast Transport Protocol
RTP	Real-time Transport Protocol
RTT	Round Trip Time
SAP	Select AP
SMTP	Simple Mail Transfer Protocol
SND_ACK_TOME	Send ACK To Me
SPM	Source Path Message
SRM	Scalable Reliable Multicast
ST	Shared Tree
TCP	Transmission Control Protocol
TMTP	Tree-Based Multicast Transport Protocol
TRACK	Tree-Based ACK
TRAM	Tree-based Reliable Multicast Protocol
TRM	Transport protocol for Reliable Multicast
TTL	Time To Live
UDP	User Datagram Protocol
XTP	Xpress Transport Protocol

CHAPTER 1

INTRODUCTION

IP multicasting is a method for transmitting the same data packets to multiple receivers. It refers to sending the same information to a group of receivers instead of a single receiver such as in unicasting or all of the receivers in the network such as in broadcasting. Distributed database updates, audio and video conferencing, distance learning, transmitting stock quotes to multiple brokers and network games are some examples for multicast applications.

IP Multicasting is a network-efficient technique for distributing information to a large group of receivers. In IP multicasting a sender can transmit a single copy of each packet without knowing who will receive it. Bandwidth consumption is minimized since only a single copy of a multicast packet flows over each link and intermediate routers. Responsibility of group management is on the receivers instead of the sender, since it is difficult for a sender to maintain the size and membership state of a growing and frequently changing group. An end-to-end control mechanism, like TCP in unicast transmissions, is not applicable for IP multicasting. Because feedbacking of all control information from receivers to a single sender causes a burst of traffic towards the sender. Therefore a specialized transport layer protocol must be defined to deal with the challenges of the end-to-end reliability, congestion, flow control and scalability problems.

Reliability issue of multicasting contains the challenges for detection and recovery of packet losses and for ordered delivery of the entire data. The basic definition of Reliable Multicast is delivering all data units to all receivers in a multicast group. In detail, reliability can be analyzed under three different concepts, namely total reliability, semireliability and time-bounded reliability. Total reliability guarantees error-free delivery of all data to all receivers. In this kind of applications all other data packets are useless if at least one packet is lost. Semireliability supports retransmission of lost packet or some error correction codings but does not guarantee totally error-free delivery of data. It is used by loss-tolerant real-time applications. Time-bounded reliability is also used by real-time applications but with strict jitter requirements. In which retransmission of a lost packet is only performed for up to a certain time.

In literature, there have been some proposals for the taxonomy of reliable multicast protocols. In this study we investigate these different classifications and select one of them for our comparative study. One representative protocol for each group of selected classification is selected and simulations for different test cases are applied using these protocols. The selected protocols are SRM, PGM and RMTP. Implementations for SRM and PGM previously exist in ns-2, our discrete event simulation tool, but not for RMTP. In the first stage of the work, RMTP is implemented for ns-2.

In simulation study, some performance metrics are defined for comparing the representative protocols. These performance metrics are distribution delay, recovery latency, request overhead on routers or links and repair overhead on routers or links. Test cases are generated by varying the number of multicast group members in the network, the network diameter (i.e. average end-to-end delay of the whole network) and the average loss rate of the links.

The organization of the thesis is as follows;

Chapter 2 gives a brief background for IP Multicasting. In this chapter, definition of multicasting is given and the main concepts like multicast addresses, group management and multicast routing is presented. Then, some challenging issues of IP Multicast are defined briefly.

Chapter 3 focuses on the reliability issue of IP Multicasting. It gives different definitions for reliability and analyses the methods used for error detection and recovery. Different classifications proposed until now are presented by figures and explained briefly. The representative protocols selected for simulation study are defined in detail. A detailed literature survey is also presented in this chapter.

In Chapter 4, Ns-2 implementation of RMTP is explained. The details of the protocol agents and the operation steps of the whole protocol are presented in this chapter.

Chapter 5 contains the comparative study of the thesis. In this chapter, the selected network and application model, protocol parameters and performance metrics are explained. The design of the test cases and simulation results are presented.

Chapter 6 summarizes the thesis and concludes with comments on the simulation results and on the performance of the protocols and states some future work directions.

CHAPTER 2

IP MULTICAST BACKGROUND

2.1. Definition

In IP-based networks, data can be disseminated in three district ways, namely unicasting, broadcasting and multicasting. Unicast transmission is the basic one-to-one communication i.e. a single application communicates across a network or internetwork with another application. Many of the widely used Internet applications like HTTP, SMTP, FTP or Telnet use unicast transmission. Broadcasting is one-to-all communication where the data is sent from a single sender to all other nodes within the network. Therefore for wide area network applications broadcasting does not make much sense. More localized level applications such as ARP (Address Resolution Protocol) uses broadcast communication [1].

Multicasting stands somewhere between unicasting and broadcasting. It refers to sending the same information to a group of receivers instead of a single receiver such as in unicasting or all of the receivers in the network such as in broadcasting. Multicast communication is called one-to-many or many-to-many in case of more than one sender is supported by the protocol. Updating distributed databases, audio and video conferencing, distance learning, transmitting stock quotes to multiple brokers and network games are some examples for multicast applications [2] [3].

In multicasting, senders send each data packet once and at most one copy of the packets flows through the physical links under normal conditions. For example, assume that a sender, S, wants to send a message to receivers R_1 and R_2, as shown in Figure 2.1. In case of unicast transmission, S should transmit the same data twice and the bandwidth usage between the sender and the intermediate node is doubled. In broadcasting, other receivers like R_3 will get the packets although it is not relevant with the message sent, causing unnecessary bandwidth consumption. But in multicasting, only a single copy of the message is transmitted from the sender and it is copied at the intermediate node to be sent to the multicast group. A multicast group can range in size from a few nodes to several thousands. In the example given in Figure 2.1, the multicast group consists of nodes R_1 and R_2.

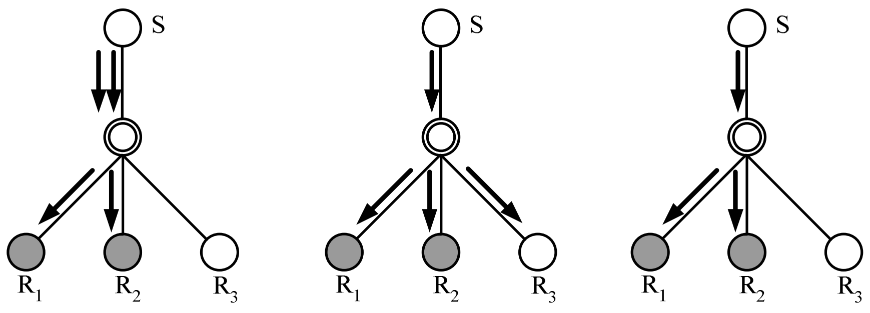

Figure 2.1 Unicast, broadcast and multicast transmissions

2.2 IP Multicast Concepts

In order for a host to participate in a multicast session, there are three main features. These are the “multicast group address” specifying that session, the “group membership protocol” for informing the nearest multicast capable router that the host wants to join or leave the multicast session and the “multicast routing protocol” to build a multicast delivery tree in which the sender is the root node and the group members are the leaf nodes of the tree.

2.2.1 Multicast Address

Multicast addresses specify an arbitrary group of hosts that have joined the group and that want to receive the packets sent to this group. Internet Protocol (IP) supports multicast transmissions using Class D IP addresses. Unlike the other types of IP addresses (Class A, B and C), an IP multicast address does not have Network ID and Host ID parts. As shown in Figure 2.2, a Class D IP address contains “1110” at higher-order four bits and the remaining 28 bits are used to identify the multicast group. Therefore 2^{28} multicast addresses in the range 224.0.0.0 to 239.255.255.255 can be generated at the same time each identifying a different multicast session. But some of these addresses have been reserved for specific usages by Internet Assigned Numbers Authority (IANA). The range of addresses from 224.0.0.0 to 224.0.0.255 is reserved to be used by network protocols on a local network segment. Packets with these addresses should never be forwarded by a router; they remain local on a subnet. For example 224.0.0.1 is used to send a message to all hosts on the subnet and 224.0.0.2 is assigned to the group of all routers on the subnet. The address 224.0.0.0 is guaranteed not to be assigned to any group [4]. Some of most common reserved multicast addresses are given in Table 2.1.

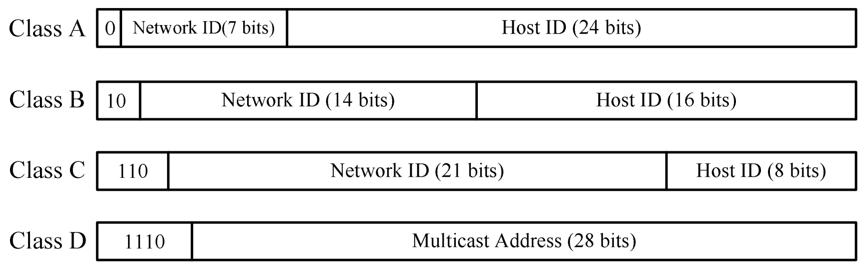

Figure 2.2 IP address classes

Table 2.1 Most common reserved multicast addresses

Address	**Function**
224.0.0.1	Broadcast to all systems on the subnet
224.0.0.2	Broadcast to all routers on the subnet
224.0.0.5	Reserved for use by OSPF routers
224.0.0.6	Reserved for use by OSPF designated routers
224.0.0.12	Reserved for use by DHCP servers

2.2.2 Group Management

A host must inform the nearest multicast-capable router that it wants to join a group. The Internet Group Management Protocol (IGMP), a standard described by RFC 1112, is responsible for managing multicast group membership on a local basis [5]. It can be thought as a multicast protocol that is run between hosts and the local router on a LAN. The local router uses IGMP to discover multicast group members on the LAN. If at least one member of a multicast group is discovered on an attached LAN segment, the router then forwards packets, sent for that group, to the LAN.

IGMP has two fundamental message types:

1. Query messages are used to discover which devices belong to a particular multicast group.

2. Report messages are used by hosts in response to queries to inform the querying device that it is a group member.

The operation of IGMP goes like this; when a host wants to join a multicast group, it sends a report message containing the multicast address of the group to address 224.0.0.2, which is the all routers reserved address. After receiving this report message a multicast capable router starts to forward the packets sent for that group to that LAN. During the multicast session, router sends periodic query messages to check at least one host who is still a group member exists in the LAN. These query messages are sent to 224.0.0.1, all host reserved address, and each group member responds with a report message. Before responding, members start a random back-off timer in order to suppress each other's reports for the same query, because at least one report message is enough to inform the router that there are one or more group members in the LAN. Routers do not have to know the number of group members. In the first version of IGMP, hosts leave the groups silently i.e. no special messages generated for leaving the group. Routers notice a difference only when the last member leaves the group, since no report message would be received responding a query message. In IGMP version 2, group members send a leave report before leaving the group. Then router issues a query message immediately to check if there is any other group member, rather than waiting for the next query time. If there is no group member in a LAN segment, router stops forwarding the packets for that group address.

2.2.3 Multicast Routing

In unicasting, routing is often treated as the shortest-path problem i.e. when two nodes want to communicate, the shortest path connecting these nodes is selected. But in multicasting, a multicast group wants to communicate each other. Now, instead of the shortest path, the minimum-weight tree spanning all of the nodes in the multicast group must be detected.

Multicast routing problem can be defined as follows [6]:

An internetwork can be modeled as a graph, consisting of a set of nodes (vertices) and a set of links (edges). Let G = (V, E) be an undirected graph, where V is the set of nodes and E the set of links. Since graph G is undirected, it models a network which has symmetric links. Let M be the multicast group including the sources. Therefore M is a subset of set V ($M \subseteq V$). Then the problem of multicast routing in communication networks is equivalent to finding a tree T in graph G such that T spans all vertices in the multicast group M. Such a tree is called multicast distribution tree.

In Section 2.1, it is mentioned that a multicast application may require one-to-many or many-to-many transmission i.e. there is only one sender per group or each group member can be both sender and receiver. Construction of the multicast distribution tree differs with respect to these types of transmission. For one-to-many transmissions, a source-specific multicast tree can be used. A source-specific tree is built from each active sender to its group [7]. As seen in Figure 2.3, a source-specific tree consists of the shortest path links between the sender and each group member. Since this type of trees are constructed for specific sources, in Figure 2.3 there are two different multicast trees for senders S_1 and S_2. Therefore the complexity of source-specific tree protocols is O(S*G), where S is the number of sender and G is the number of groups. Distance Vector Multicast Routing Protocol (DVMRP) [8] and Multicast Extension for Open Shortest Path First (MOSPF) [9] are two examples for protocols using source-specific multicast tree approach.

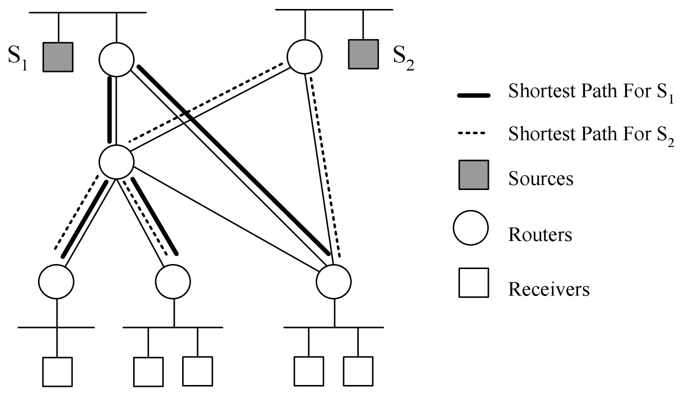

Figure 2.3 Shortest path trees in Multicast Routing

In case of many-to-many transmissions, a group-shared multicast tree must be used. Figure 2.4 illustrates a group-shared tree for the same topology and the multicast group given in Figure 2.3. In this type of multicast trees, a center node is selected and it serves as the root of the tree. It is responsible for expanding the tree when a new sender or member joins the multicast group, and for collecting traffic from all sources and multicasting it to all receivers [7]. Group-shared tree protocols has complexity O(G), where G is the number of groups. Some examples for the application of group-shared trees are Core Based Tree (CBT) [10] and Border Gateway Multicast Protocol (BGMP) [11].

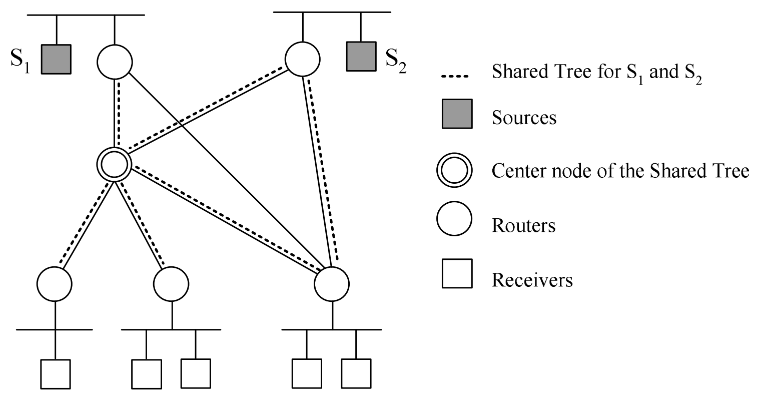
S_1
S_2
Shared Tree for S_1 and S_2
Sources
Center node of the Shared Tree
Routers
Receivers

Figure 2.4 Shared tree in Multicast Routing

2.3 Challenges of Multicasting

IP Multicasting is an efficient technique for distributing information to a group of receivers [12]. In IP multicasting a sender can transmit a single copy of each packet without knowing who will receive it. Bandwidth consumption is minimized since only a single copy of a multicast packet flows over each link and intermediate router. Furthermore, responsibility of group management (joining and leaving a multicast group) is on the receivers instead of the sender, since it is difficult for a sender to maintain the size and membership state of a growing and frequently changing group.

An end-to-end control mechanism, like TCP in unicast transmissions, is not applicable for IP multicasting. Because feedbacking of all control information from receivers to a single sender causes a burst of traffic towards the sender. Therefore a specialized transport layer protocol must be defined to deal with the challenges of the end-to-end reliability, congestion, flow control and scalability problems. This specialized transport protocol can operate over UDP (Figure 2.5.a) or be located as a transport layer protocol and operate directly with IP [13]. (Figure 2.5.b)

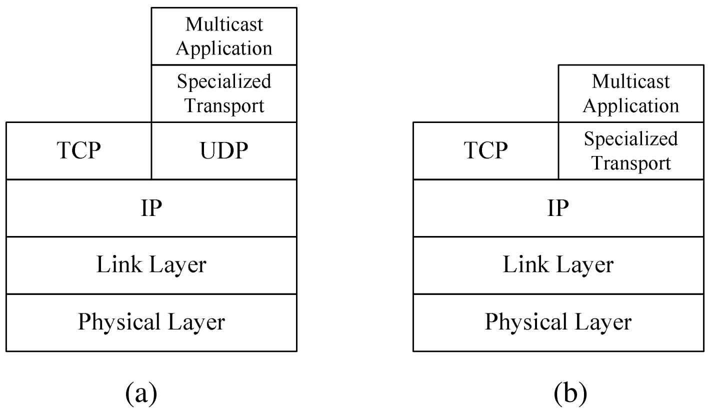

Figure 2.5 Specialized Multicast Transport (a) Over UDP (b) Over IP

These specialized transport protocols are implemented for meeting different requirements of applications. Some applications like multicast file transfer require a strict reliability, while some others may tolerate a small loss of data but need low latency, like video conferences. Some protocols deal with flow and congestion control by using multirate data transmission for heterogeneous receivers in a multicast group. Consequently, a single multicast protocol is not likely to meet the needs of all Internet applications [14]. Therefore different protocols have been implemented for different requirements. In Chapter 3, the ones related with reliability are analyzed in detail.

CHAPTER 3

RELIABLE MULTICAST PROTOCOLS

3.1 Introduction to Reliable Multicast

The basic definition of Multicast Reliability is stated as delivering all data units to all receivers in a multicast group. But this concept can be further classified into three subgroups, namely total reliability, semi-reliability and time-bounded reliability. Total reliability guarantees error-free delivery of all data to all receivers. File transfer applications are an example for applications requiring total reliability [15]. In this kind of applications all other data packets are useless if at least one packet is lost. Semi-reliability supports retransmission of lost packet or some error correction codings but does not guarantee totally error-free delivery of data. It is used by loss-tolerant real-time applications [2]. Time-bounded reliability is also used by real-time applications but with strict jitter requirements. It states that retransmission of a lost packet is only performed for up to a certain time [16]. In this study we will focus on total reliability.

For achieving reliable data transfer, two major tasks are required i.e. error detection and error recovery. Different transport layer solutions can be proposed for these tasks. These are Automatic Repeat Request (ARQ), Forward Error Correction (FEC) and sometimes combination of these two i.e. hybrid solutions.

The basic idea of the ARQ approach is to retransmit a packet only if it is lost by at least one receiver. Depending on whether error detection is done by the sender or the receivers, reliable multicast protocols could use positive (ACK) or negative (NACK) acknowledgements. When using ACKs, the sender retransmits messages until ACKs from all receivers are received. This approach does not scale well because ACKs sent by each receiver for each received packet may lead to serious network congestion (ACK implosion). In addition, the sender has to keep the state of the multicast group. Using NACKs shifts the error detection load from the sender to the receivers. Receivers transmit NACK packets only when a packet loss is detected. In order to reduce the implosion problem, different NACK suppression mechanisms could be applied, since the sender only needs to know that at least one receiver is missing data.

FEC mechanism consists in sending redundant packets together with original data packets. For every *k* original data packets, *h*=*n*-*k* parity packets are constructed. All *k* data packets can be reconstructed if any *k* packets out of *n* are correctly received. The FEC-based approach reduces the end-to-end latency compared to the ARQ, since receivers do not have to wait for the retransmission of lost packets. But this is at the cost of bandwidth since redundant packets are sent.

In addition to ARQ and FEC there are two hybrid methods combining FEC with ARQ, namely the layered approach and the integrated approach [17]. The layered approach considers FEC as an independent layer below the ARQ-based protocol. The advantage of such a solution is that FEC is transparent to the ARQ protocols and transparently improves ARQ performances. Besides, if an application does not require total reliability, the ARQ protocol may be skipped in order to only use the FEC layer. In the integrated approach, original data packets are transmitted with or without parity packets. A receiver will request more parity packets when it detects packet losses. In this method, FEC and ARQ operate in the same layer, as part of the same protocol.

3.2 Classification of Reliable Multicast Protocols

There are various reliable multicast protocol classifications in the literature. Some of them group protocols according to the application requirement, while some others use the way that the protocol recovers packet losses or whether it uses router assistance. In this section, different classification methods are investigated briefly and a figure showing the location of well known protocols for each classification is given.

In [18], multicast transport protocols are classified according to the application characteristics, namely General Purpose Protocols, Protocols for Multipoint Interactive Applications and Protocols for Data Dissemination. General Purpose Protocols are defined as message-oriented and not designed for a specific application. Second group of protocols are the ones that support Multipoint Interactive Applications. These are implemented for many-to-many transmissions and generally used by real-time applications. Protocols for Data Dissemination are designed for delivering same data to multiple receivers in a non-real-time manner. These protocols have a strict reliability requirement compared to other type of protocols. Figure 3.1 gives the classification according to application characteristics and lists some example protocols for each group.

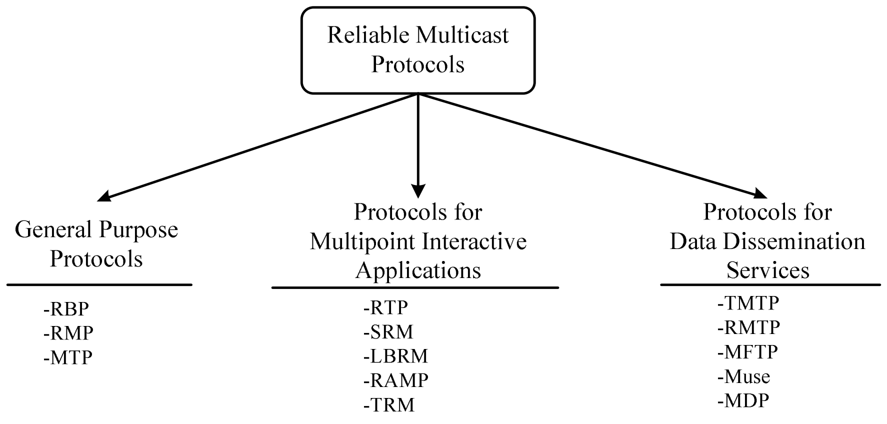
Reliable Multicast Protocols
General Purpose Protocols
-RBP
-RMP
-MTP
Protocols for Multipoint Interactive Applications
-RTP
-SRM
-LBRM
-RAMP
-TRM
Protocols for Data Dissemination Services
-TMTP
-RMTP
-MFTP
-Muse
-MDP

Figure 3.1 Classification of Reliable Multicast Protocols - 1

Another way of classification is distinguishing the protocols according to the organization of retransmissions. In [19] and [20], reliable multicast protocols are analyzed under four groups; Sender-initiated, Receiver-initiated, Tree-based and Ring-based reliable multicast protocols. In the sender-initiated protocols, each receiver sends a unicast ACK to the sender for each packet that the receiver obtains correctly. The sender maintains the state of all receivers to whom it has to send packets and from whom it has to receive ACKs. Transmissions or retransmissions are multicast to all receivers. But this mechanism suffers from the ACK-implosion problem. Therefore receiver-initiated protocols have been proposed. In this approach, a receiver only sends a negative acknowledgment (NACK) when it detects an error or a lost packet.

However, still in some cases a burst of NACK packets may cause problems at sender network. Therefore transmission of control packets must be organized in a way that ACK/NACK implosion is prevented. In tree-based reliable multicast protocols, the whole multicast delivery tree is divided into subtrees and all control packets are transmitted within this subtree. Only one ACK or NACK packet is transmitted to the upper node by the root node of the subtree. On the other hand, in Token-Ring based protocols, there is only one token site responsible for collecting ACK/NACK packets and transmitting towards the sender. The token is periodically passed to the next node of the ring. In Figure 3.2, some example protocols for each group are listed.

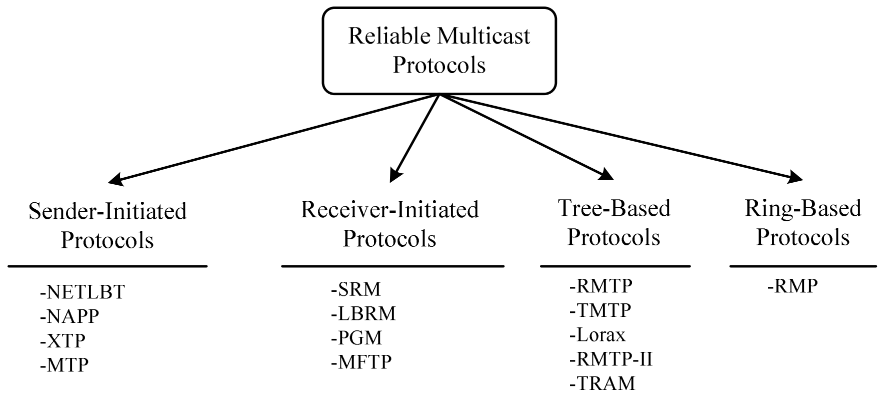

Figure 3.2 Classification of Reliable Multicast Protocols - 2

[21] classifies the protocols in a very similar way to the previous one. But it defines two main groups as Sender-initiated and Receiver-initiated. Other groups are located under Receiver-initiated protocols as seen in Figure 3.3. These subgroups are Cloud-based, Tree-based and Ring-based reliable multicast protocols. Cloud-based multicast protocols are defined as the protocols that do not require the receivers to be arranged in a definite structure. These protocols do not maintain the membership information that accounts for their good scalability. Tree-based and Ring-based protocols are defined as in previous classification. Figure 3.3 lists some example protocols for this classification.

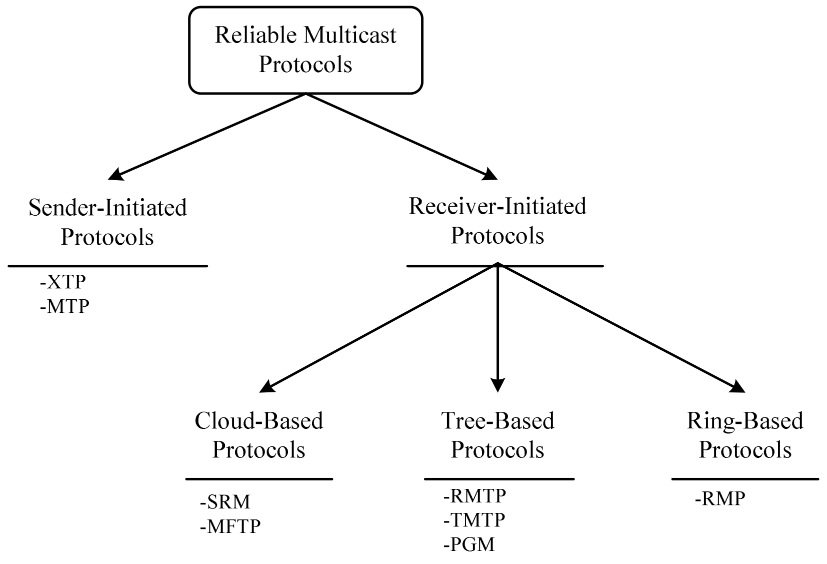

Figure 3.3 Classification of Reliable Multicast Protocols - 3

A newer classification adds router-assistance and forward error correction methods to the taxonomy. In [16] and [22], reliable multicast protocols are divided into four groups; namely NACK-only protocols, Tree-based ACK (TRACK) protocols, Router assisted protocols and Open Loop protocols, as given in Figure 3.4. NACK-only protocols attempt to limit traffic by only using NACKs for requesting packet retransmission. They do not require network infrastructure.

TRACK protocols use ACKs. In order to avoid ACK implosion, ACKs are suppressed in a tree shaped infrastructure.

Router assisted protocols also use negative acknowledgments for packet recovery. These protocols take advantage of router software to do constrained negative acknowledgments and retransmissions. Router assisted protocols can also provide other functionalities like congestion control. Open loop protocols use sender-based Forward Error Correction (FEC) methods with no feedback from receivers or the network to ensure good throughput.

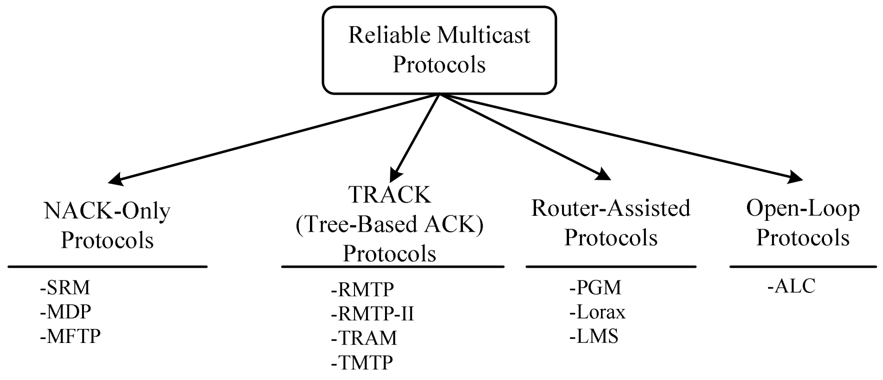

Figure 3.4 Classification of Reliable Multicast Protocols - 4

3.3 Literature Survey

[23] defines the criteria for an ideal reliable multicast protocol as reliability, efficiency and tolerance. Reliability is guaranteeing the file to be delivered entirely to all receivers. Being efficient means both the total number of packets each client needs to receive and the amount of time required to process the received packets to reconstruct the file should be minimal. Being tolerant means that, the protocol should tolerate a heterogeneous population of receivers, especially a variety of end-to-end packet loss rates and data rates.

For achieving these criteria, many reliable multicast protocols have been proposed. Most of these protocols suffer from the feedback implosion problem which occurs when a large amount of receivers send feedback packets to sender synchronously. To overcome this problem, different approaches have been proposed. One of these is using hierarchical ACKs. In [24] Reliable Multicast Transport Protocol (RMTP) organizes the group members into a hierarchical control tree, which governs feedback propagation and processing. Intermediate nodes in the control tree, or designated receivers (DRs) are responsible for buffering data received from the source, processing ACKs from their children, and retransmitting lost packets. Therefore, DRs provide local recovery. Protocol uses window-based flow control and defines a maximum transmission rate set at group establishment time. RMTP allows dynamic group membership: receivers may join and leave a multicast group any time during a session, and it is guaranteed that they receive the entire data reliably. This feature comes at the price of having the DRs buffer transmitted data during the whole session.

[25] proposes another hierarchical ACK protocol, Tree-Based Multicast Transport Protocol (TMTP). For error and flow control, TMTP organizes group members into a hierarchy of subnets or domains. Typically, all the group members in the same subnet belong to a domain and a single domain manager acts as a representative of the domain. The domain manager is responsible for recovering from errors and handling local retransmissions if one or more of its children do not receive some packets. The domain managers are organized in the form of control tree. The sender serves as the root of the tree and has at most K domain managers as children. Each domain manager will accept at most K other domain managers as children, resulting in a tree with maximum degree K. The degree of the tree (K) limits the processing load on the sender and the internal nodes of the control tree. Like RMTP, TMTP uses a window-based flow control and the maximum transmission rate is defined at group creation. It also supports dynamic group membership.

Another approach for avoiding feedback implosion is NACK-based protocols. [26] defines the Scalable Reliable Multicast (SRM), where group members multicast NACKs to request retransmission of a lost packet, which can be answered by any member that has the packet. To avoid generating multiple copies of retransmitted data, retransmissions are multicast to the group. To further reduce the multiple copy problem, a member waits a random period of time before sending a NACK or retransmitting data, and suppresses its own transmission in case it hears it from another member of the group. According to [27], for a large number of receivers spanning wide-area networks, NACK suppression alone is not very efficient. SRM does not specify any congestion control mechanisms.

[28] proposes another NACK-based protocol, Multicast File Transfer Protocol (MFTP). The protocol consists of two parts; an administrative protocol to set up and tear down groups and sessions, and a data transfer protocol used to send the whole file reliably to the multicast group. Using the administrative protocol, the MFTP sender announces a file transfer session periodically during the announcement phase. In response to announcements, clients register to the multicast group to receive the file. MFTP sends data in passes. In the first pass, the sender sends a block of packets and collects the NACKs for that block from all receivers. These NACKs are logically OR-ed together to represent the collective need for repairs for the receiving group. These repairs are sent by the sender in a second pass to the group. Receivers already have the repair simply ignore the packets. Holes in the data due to the packet drops are filled as the repairs are received. So the protocol does not provide packet ordering. Furthermore, since receivers can request retransmissions only at the end of the passes, MFTP is suitable only for non-real time applications.

Router support is another method for avoiding feedback implosion. The protocols that uses router support can be divided into two categories. Protocols in the first category use router support for only directing the retransmission requests to a proper replier. Lightweight Multicast Service (LMS) [29] is an example for this category. In LMS, each router selects one of its downstream links as the replier link. After detecting a loss, a receiver will send a retransmission request to a nearby router. On receiving a retransmission request, a router will redirect it to the replier link if it comes from other downstream links, or forward it to an upstream router if it comes from the replier link. Thus, only one retransmission request is sent upstream from a router for a certain lost packet and feedback implosion is reduced. LMS does not function well in case of failure of the repliers directly above or under the link where the loss occurs or in bidirectional shared routing trees. For these problems, Search Party [30] and Reliable Multicast for Core-based Multicast (RMCM) [31] have been proposed.

Second kind of router assisted protocols uses routers for feedback aggregation or suppression and for local recovery. Pragmatic General Multicast (PGM) [27] uses PGM-capable network elements (NEs), i.e. routers enhanced to support PGM, for NACK elimination and suppression. All NACKs are transmitted to the sender by aggregating them at the intermediate NEs. Therefore the sender is responsible for replying all retransmissions. The congestion control approach of PGM uses NACKs for rate adjustment. But in [32], it is shown that this congestion control approach does not scale well for large multicast groups.

In [33], the authors propose a different method for achieving reliability, called Semantically Reliable Multicast. The model is based on the concept of message obsolescence. An obsolete message is defined as the one whose content or purpose is superceded by another message. This knowledge is used by the protocol to selectively discard some messages from buffers in the presence of overload conditions. By allowing obsolete messages to be discarded, the system better tolerates the occurrence of performance perturbations without demanding additional resources. In this work, it is seen that applications requiring high throughput exhibit message obsolescence and semantically reliable multicast protocols result in an improvement of throughput stability. The implementation and configuration parameters are also analyzed to see their effects on the performance of semantic reliability.

In [19], authors compared four different groups of reliable multicast protocols which are classified according to the method given in Figure 3.2. They stated that sender-initiated protocols are not scalable because the source must listen every receiver. Receiver-initiated protocols are far more scalable, unless NACK avoidance schemes are used to avoid overloading the source with retransmission requests. However, because of the unbounded-memory requirement, this protocol class can only be used efficiently with application-layer support, and only for a limited set of applications. They showed that ACK trees are a good answer to the scalability problem for reliable multicasting. Because tree-based protocols (e.g. RMTP) distribute the responsibility of retransmission to receivers and they employ techniques applicable to either sender-initiated or receiver-initiated protocols within local groups of the ACK tree. Mechanism that can be used with all the receivers of a session in a receiver-initiated protocol can be adopted in a tree-based protocol, with the added benefit that the throughput and number of supportable receivers is completely independent of the size of the receiver set.

[34] presents a comparison of SRM with Bimodal Multicast, which is a reliable multicast protocol proposed by the authors. According to this study, when a network has lossy links, even if the loss rate is low, SRM can generate very high rates of request or repair overhead, due to the requests for retransmissions of data, sent using multicast and hence seen by significant numbers of processes, and repair messages, also sent using multicast. They concluded that as the network grows larger, the absolute rate of packet losses increases. These take the form of duplicate requests and duplicate repairs. Beside this, a significant percentage of SRM packets experience long delays, and many applications would thus be forced to buffer very large amounts of data. For applications in which data freshness is at all important, this would seem to be a real drawback for the protocol.

In [35], authors compared three reliable multicast protocols, namely SRM, MFTP, MFTP/EC, by using Network Simulator-2. They showed that, SRM floods the network with repair packets substantially when loss rates are high, resulting in even higher loss rates due to additional traffic. According to the simulation results, every lost packet triggered 2.27 request packets and 2.33 repair packets on average in low a network traffic condition. In high traffic case, lost packets need an average of 3.56 request packets and 3.62 repair packets for recovery.

In literature, there are hardly any comparative performance studies for RMTP. [36] investigates the delay characteristics of some reliable multicast protocols. According to this study, average delivery delay of RMTP is almost independent of group size and sending rate of the source. [37] also presents a comparative delay analysis of tree-based reliable multicast protocols, namely RMTP and TMTP. It is shown that tree-based ACK (TRACK) protocols provide low delays and good scalability compared to nonhierarchical approaches. NAK-based protocols achieve the best scalability but TRACK protocols (i.e. RMTP) achieve the lowest delays.

3.4 Related Work

In this section, the protocols that are selected for our simulation study, namely RMTP, SRM and PGM are discussed in detail. Design concepts of the protocols like recovery methods, protocol parameters, and network and application models that the protocols uses are evaluated.

3.4.1 Reliable Multicast Transport Protocol (RMTP)

The Reliable Multicast Transport Protocol (RMTP) [24] is a one-to-many, tree based positive acknowledgement protocol which provides sequenced and lossless file transmission. It uses IP multicast to forward the multicast packets down on a delivery tree to all group members and provides reliability by using a packet based selective repeat retransmission scheme. To avoid ACK implosion problem, the protocol divides the whole multicast into hierarchical local regions and assigns a Designated Receiver (DR) to each local region. A DR is a special receiver which assists the sender in processing ACKs and retransmitting data. Periodic ACK packets are unicasted to the local DR and DRs send the ACK packets to the high level DR, until the DRs in the highest level of the multicast tree send ACK packets to the sender.

RMTP sender divides the data to be transmitted into fixed-size data packets, except the last one. A data packet is identified by packet type DATA and DATA_EOF identifies the last data packet. All of the packet types used in RMTP are given in Table 3.1.

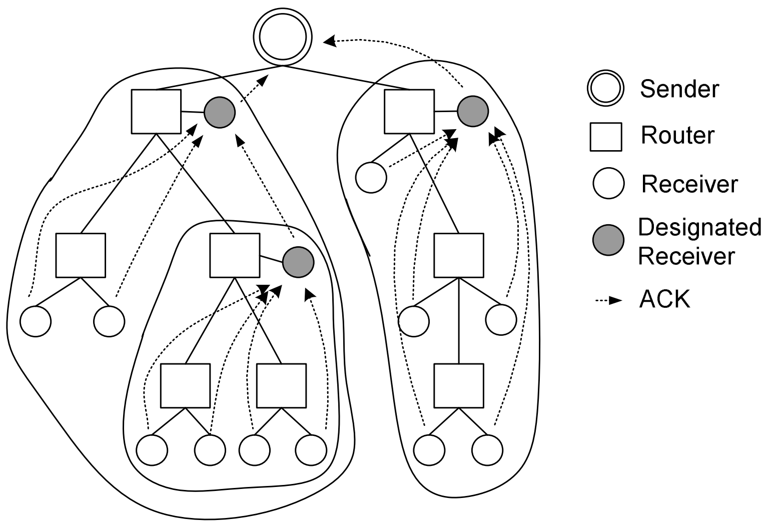

Figure 3.5 Local regions in RMTP

Table 3.1 Packet types of RMTP

Packet Type	Definition
ACK	ACK packet
ACK_TXNOW	ACK - immediate transmission request
DATA	Data packet
DATA_EOF	Last data packet
RESET	Packet to terminate a connection
RTT_MEASURE	Packet to measure round-trip time
RTT_ACK	ACK to RTT_MEASURE packet
SND_ACK_TOME	Packet for selecting an AP

RMTP achieves scalability by three important design features [24]. First, the state information maintained at each multicast group member is independent of the number of members. Therefore, joining or leaving of a group member does not affect the state information of sender or other members. Second, RMTP places the responsibility of sequenced and lossless delivery on the receivers. So the sender does not have to deal with status of each receiver. Third, the concept of using a DR of each local region distributes the responsibility of processing ACKs and performing retransmissions among the sender and several DRs.

In RMTP, the sender assigns sequence number to all packets starting from 0 and multicast the packets to the whole group. Receivers are responsible for detecting packet losses by analyzing the sequence numbers of incoming packets. A gap at the sequence numbers means a loss and the receiver should request the retransmission of that packet. Receivers send periodic ACK packets to inform their local status to their local DRs. Each ACK packets contains a sequence number L and a bitmap V. Sequence number L indicates that the receiver has correctly received all packets with a sequence number less than L. A 0 in bitmap V means a packet loss and a 1 indicates the successful reception of the packet. Therefore, an ACK packet carries both positive and negative acknowledgment.

ACKs are periodically unicast to DR. A DR also responds to the request periodically but it may be either unicast or multicast transmission. T_{retx} is the connection parameter which determines the retransmission interval of a DR. During the T_{retx} interval, DR collects the ACK packets from its local region. If retransmission request for a data packet exceeds $MCAST_{thres}$ parameter, then DR multicasts the repair packets to its local region. Otherwise, it unicasts the packets to the receivers who demand that transmission. T_{retx}, $MCAST_{thres}$ and other connection parameters are given in Table 3.2.

When a connection between the sender and a group of receivers is established and the receiver has received the first packet of the session, it starts to generate periodic ACK packets at T_{ack} interval. T_{ack} is a dynamic parameter which is adjusted based on a round-trip time measurement between the receiver and its Acknowledgement Processor (AP). AP is a member who processes ACK packets. So, it may be the sender or a DR.

Table 3.2 Connection parameters of RMTP

Parameter	Definition
W_r	receive window size in packets
W_s	send window size in packets
T_{dally}	delay after sending the last packet
T_{retx}	time interval to process retransmission requests
T_{rtt}	time interval to measure RTT
T_{sap}	time interval to send SND_ACK_TOME
T_{send}	time interval to send data packets
Packet Size	data packet size in octets
Cache Size	sender's in-memory data cache size
$CONG_{thresh}$	congestion avoidance threshold
$MCAST_{thresh}$	multicast retransmission threshold

In order to terminate the connection, the sender uses a timer. After sending the last packet, the sender starts a timer and waits for T_{dally} seconds. If the sender receives an ACK packet before the timer expires, it resets the timer to its initial value. Otherwise, it assumes that all receivers have received all packets and terminates the connection i.e. deletes all the state information about the connection. In addition to that, a connection can be terminated by a RESET packet in an unexpected case.

RMTP uses a window-based flow control scheme. Sender has a window of W_s packets, which indicates the maximum numbers of packets that the sender can transmit without receiving ACK for them. Each receiver has a receiving window of W_r packet, where W_r is the buffer size of the receiver. Receivers keep the incoming packets at the buffer and deliver them to the application in sequence. In a RMTP session, the sender initiates the data transmissions at interval T_{send}. Therefore sender's maximum transmission rate measured over a transmission interval can be adjusted by the connection parameters T_{send}, W_s, Packet_Size as follows:

$$MaximumTransmissionRate = \frac{W_s \times Packet\_Size}{T_{send}}$$

RMTP uses a slow-start congestion avoidance mechanism in order to avoid sending new packets to a congested network. The sender uses a congestion window (con_win) to limit the number of packets being sent during transmission period, T_{send}. During T_{send} it computes N, the number of ACK packets which contain a retransmission request. If N exceeds a connection parameter, $CONG_{thres}$, it sets cong_win to 1. Since the sender selects the minimum of current window size and cong_win as the usable send window, setting cong_win to 1 reduces data transmission to one packet per T_{send}. If N is less than $CONG_{thres}$ at the end of a T_{send} interval, the sender increases cong_win by 1, until cong_win reaches W_s.

RMTP receivers send periodic ACK packets with a period of T_{ack} seconds. Sending ACKs too frequently may cause APs to retransmit the same repair packet without knowing the first packet was received by the receivers. Therefore T_{ack} is computed using the round-trip time (RTT) between the receiver and its AP. In order to measure RTT, the protocol defines two packet types, namely RTT_MEASURE and RTT_ACK. A receiver sends RTT_MEASURE packets to its AP at a fixed interval, T_{rtt}. Each RTT_MEASURE packet contains a timestamp indicating the time it is transmitted. When a AP receives a RTT_MEASURE packet it immediately changes the packet type to RTT_ACK and sends it back to the receiver. Receiver computes the RTT using the time it received the RTT_ACK packet and the time stamp stored in the packet.

Since a DR may fail during a session, RMTP has a mechanism for selecting the nearest DR as AP by a receiver. For this purpose, the sender and all DRs multicast a SND_ACK_TOME packet to their subtrees with a period of T_{sap} seconds. Each SND_ACK_TOME packet contains an identical time-to-live (TTL) value. Since each router decrements the TTL value while transmitting the packets, the SND_ACK_TOME packet within the highest TTL value is the one, which has been generated by the nearest DR. The receiver selects that DR as its AP and stores its TTL value. If it receives a SND_ACK_TOME packet with a TTL value greater than the stored one, it changes its AP and stores the new TTL value. Initially sender is the AP for all receivers. But since the receivers store a TTL value of 0 at the beginning of the session, they can select the nearest DR as AP immediately after the connection established.

In [24], authors present the results of the experiment which was performed on the Internet by a prototype implementation of the protocol. The test setup contains 18 multicast receivers located at five geographic areas, namely five different campus networks. In each experiment, a 1 Mb file was multicast to the receivers, and same experiments were repeated at different times of the day, in order to observe the response of the protocol against different network traffics. The results of the experiments indicate that receivers, regardless of their geographic location, take about the same time to correctly receive the file, which shows that RMTP is able to adapt to receivers in various network environments. But as a result of this, in a heterogeneous environment, slow receivers and links with low bandwidth limit RMTP's performance [24]. Since there is not enough information about the propagation delay, bandwidths and traffic characteristics of the experiment performed, we could not simulate the same test cases for comparing the results of a real experiment and our protocol implementation. Instead of this, we generated a new test setup and measured the performance of the protocol on this setup, whose details will be presented in Section 5.1.

3.4.2 Scalable Reliable Multicast (SRM)

Scalable Reliable Multicast [26] is a NACK-based reliable multicast protocol designed for a Many-to-Many application, using IP Multicast mechanism for packet transmission. SRM employs a "decentralized error recovery"; that means, a node that detects a packet loss, via a gap in the packet sequence, multicasts a NACK for that packet, and any node that has the data packet can multicast a repair packet for that loss. In other words, the repair process has been distributed among all group members.

In SRM, when a new packet is generated, it is multicast to the group. Actually all data packets defined in SRM, namely the original data, request packets and repair packets are multicast to the whole group. Since a repair request is multicast to the whole group, more than one receivers who have the desired packet can reply simultaneously. In order to prevent such duplicate retransmissions, SRM has two mechanisms:

i. Use a delay of some random interval before transmitting a request/repair packet (back-off),
ii. Suppress transmission of a request/repair packet if someone else has already sent the same packet (suppression).

The timers that receivers use before transmission are adjusted based on the distance between the sender and the source. To measure these distances, group members send low-rate, periodic session messages. By using these messages, a group member can learn the group membership, measure the delay among group members and detect the sequence number of the last packet sent in the session. Session messages are designed to take only 5% of the traffic in the session. [26]

Receivers decide a packet loss by detecting a gap in the sequence numbers of the incoming packets. Therefore the sequence number of the last packet is necessary for detecting the drop of the last packet. Session messages contain a Source-ID and a timestamp. The timestamps are used to estimate the distance between group members.

Loss Recovery: A receiver who has a missing data waits for a random time before sending repair request, in order to detect whether another member sent a request for the same data. If a request for the same loss reaches during the back-off time it cancels the repair request. If there is no repair request or retransmission when the back-off timer expires, the member multicast a repair request to the whole group. The random back-off timer is set to a value which is a function of the member's estimated distance to the source of the original packet. It is chosen from the uniform distribution on

$$[C_1 d_{S,A} , (C_1+C_2) d_{S,A}]$$

seconds where $d_{S,A}$ is member A's estimated distance to the original source, S, of the missing data. The numbers C_1 and C_2 are parameters for the request algorithm.

When a group member receives a request for a packet that it has already received, it starts a back-off timer in order to prevent duplicated retransmissions. If no retransmission reaches during this period, it multicast the repair packet to the whole group. Repair timer is set to a value from the uniform distribution on

$[D_1 d_{A,B} , (D_1+D_2) d_{A,B}]$

seconds, where $d_{A,B}$ is host B's estimate of one-way delay to host A, and the numbers D_1 and D_2 are parameters of the repair algorithm. Figure 3.6 illustrates the loss recovery in SRM.

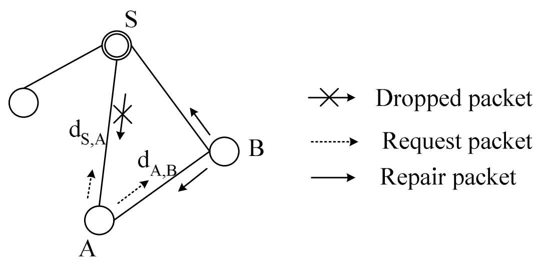

Figure 3.6 Loss recovery in SRM

3.4.3 Pragmatic General Multicast (PGM)

Pragmatic General Multicast (PGM) [27], [38] is a reliable multicast transport protocol for applications that require multicast data delivery from a single source to multiple receivers. PGM runs over a best effort datagram service, such as IP multicast. It obtains scalability via hierarchy, forward error correction, NACK elimination and NACK suppression. Hierarchy is supplied by using PGM-capable network elements (NE) i.e. the routers enhanced to support PGM in addition to IP multicast.

A “session" of the PGM protocol (a given data transfer from a source to a group of receivers) builds a tree: the source is the root of the tree, the receivers are the leaves, and the other network elements are intermediary nodes. This tree may change during the session by the dynamic join/leave of receivers. PGM has five different types of data packets;

ODATA: Original Data packets,

RDATA: Repair packets,

NACK: Negative Acknowledgements, generated by receivers in case of a packet loss,

NCF: NACK Confirmation packets, sent by a network element or source when a NACK received,

SPM: Source Path Message, control messages sent in order to maintain the routing information and state of the source of a session.

Figure 3.7 shows a distribution tree and the direction followed by the five basic packet types of the protocol.

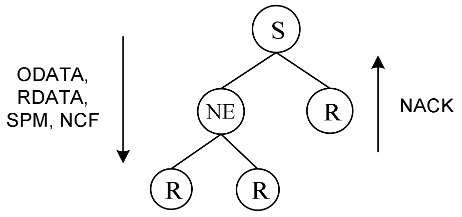

Figure 3.7 Distribution tree of the PGM with packets involved
(S = source, NE = network element, R = receiver)

In the normal data transfer, a source multicasts sequenced data packets (ODATA) along the distribution tree to the receivers by using IP multicast. When a receiver detects missing data packets from the expected sequence, it unicasts periodic negative acknowledgments (NACKs) containing the sequence number of missing data to the last network element of the path. Network elements forward NACKs hop-by-hop to the source using the reverse path, and confirm each hop by multicasting a NACK confirmation (NCF) in response to the child from which the NACK was received. Receivers and network elements stop sending NACK at the reception of a corresponding NCF or RDATA. Finally, the source itself receives and confirms the NACK by multicasting an NCF to the group. Then the source generates repairs (RDATA) in response to the NACK.

To avoid NACK implosion, PGM specifies procedures for NACK elimination within network elements in order to propagate just one copy of a given NACK along the reverse path of the distribution tree. The protocol also has a NACK suppression mechanism; receivers wait for a random time before delivering NACKs. If a RDATA or another NACK which has the same content is received during this period, the receiver cancels its own NACK.

PGM uses periodic SPMs (Source Path Messages) to improve the data transfer operations. SPMs have two functions. First, they carry routing information used to maintain up-to-date PGM neighbor information and a fixed distribution tree. Second, they complement the role of data packets when there is no more data to send by holding the state of the sender window. In this way, the receiver may detect data losses and send further NACKs.

PGM has three basic components, each having different functions. These components are source, receiver and network element.

Source functions: The source executes five functions; multicast of ODATA packets, multicast of SPMs, multicast of NCFs in response to any NACKs received, multicast of RDATA packets, and maintain (update and advance) of the transmit window. The transmit window plays an important role in the PGM operations. Any information produced by the application using PGM (upper level in the network layers) is put in the transmit window and split in several ODATA packets, numbered circularly from 0 to 2^{32} - 1. This data is maintained in the window TXW_SECS time units for further repairs and sent with a maximum transmit rate of TXW_MAX_RTE (bytes/seconds). The left edge of this window, TXW_TRAIL, is defined as the sequence number of the oldest packet available for repairs. The right edge, TXW_LEAD, is defined as the sequence number of the most recent data packet the source has transmitted. To provide information about the sender window, TXW_TRAIL edge is sent with O/RDATA and SPM packets and the TXW_LEAD edge is included only in SPMs. If TXW TRAIL = TXW LEAD + 1, the window is considered empty. The edge TXW_LEAD is advanced when data is produced by the application.

Two types of SPMs are sent by the source: ambient SPMs are sent to maintain routing information; heartbeat SPMs are transmitted in absence of data at a decaying rate, in order to assist detection of lost data before the advance of the transmit window.

Receiver functions: The receiver executes four functions; receive ODATA and RDATA within the transmit window and eliminate duplicates, unicast NACKs repeatedly until it receives a matching NCF if it detects a loss, suppress NACKs sending after the reception of the NCF, maintain a local receive window.

The receive window is determined entirely by the packets from the source, since it evolves according to the information received from the source (data packets and SPMs). For each session, the receiver maintains the buffer and the two edges of the window: RXW_TRAIL is the sequence number of the oldest data packet available for repair from the source (known from data and SPMs) and RXW_LEAD is the greatest sequence number of any received data packet within the transmit window.

Network element functions: Network element forwards ODATA without intervention. They play an important role in routing, NACK reliability, and avoiding NACK implosion. They forward only the first copy of a NACK (Constrained NACK Forwarding) and discard NACKs for which they have repair data (NACK Elimination). They also forward RDATA only to the child which signaled by a NACK the loss of the corresponding data (Constrained RDATA Forwarding).

CHAPTER 4

NS-2 IMPLEMENTATION OF RMTP

RMTP does not exist as a reliable multicast protocol option in the latest version of Network Simulator version 2.27, which was released in January 2004. Therefore, in this study ns simulation code for RMTP has been implemented and appended to ns version 2.27. This section explains the details of the implementation.

The implementation introduces a new RMTP packet type, and three new agents namely; sender agent, receiver agent and DR (Designated Receiver) agent. RMTP packet header and each of the agents are implemented as a C++ class and operate with some associative classes like timer classes, Tcl linkage classes and classes defining the sender and receiver windows. The interaction of classes is illustrated in Figure 4.1.

4.1 RMTP Packet Type

An RMTP packet is inherited from ns2 packet class, so it has all ordinary packet headers of ns2, like “IP header” or “common header”. Besides, it has an RMTP header indicating the subtype of the packet. There are six subtypes, which are defined by the protocol. Some subtypes have an extra header for storing the required information for the protocol. Table 4.1 gives RMTP packet subtypes and headers of each.

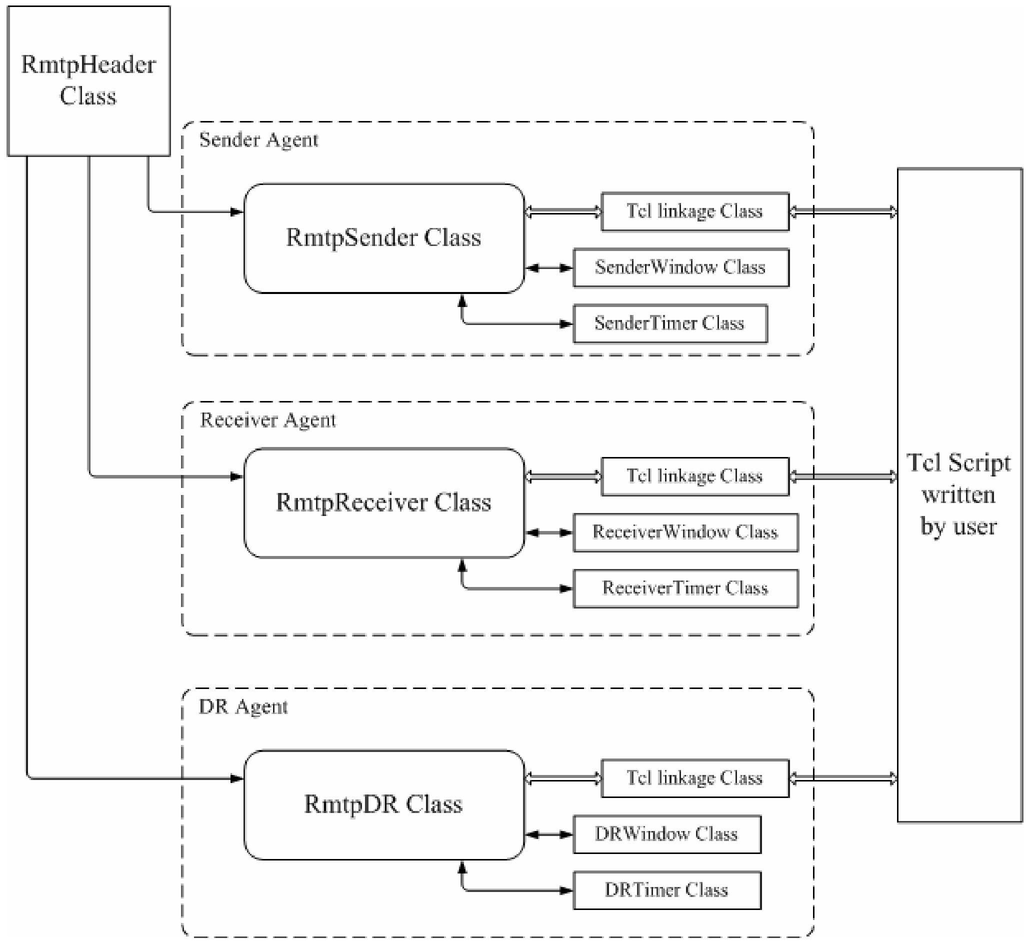

Figure 4.1 Interaction of classes

RMTP_DATA header consists of two fields, one of which indicates the type of data packet, whether DATA or DATA_EOF, and the other keeps a data ID. All agents maintain their receiver or sender windows according to these data IDs. If a data packet is lost and a retransmission request is received for the packet, the AP (sender or DR) sends a new data packet with the same data ID. Thus the receiver realizes that it receives a lost packet and updates its receiver window according to the ID.

Table 4.1 RMTP subtypes and headers

RMTP Subtype	Direction*	Headers
DATA	⇓	(RMTP header) + (RMTP_DATA header)
RTT	↑	(RMTP header) + (RMTP_RTT header)
ACK	↑	(RMTP header) + (RMTP_ACK header)
RTT_ACK	↓	(RMTP header) + (RMTP_RTT_ACK header)
SND_ACK_TOME	⇓	(RMTP header)

(*) Downward arrows indicate transmission from an AP to receiver(s)
Upward arrows indicate transmission from a receiver to an AP
Single lined arrows means unicast transmission
Double lined arrows means multicast transmission

RMTP_RTT and RMTP_RTT_ACK headers have a field for storing a timestamp. The time that a RTT packet sent is written to this fields and it is used for calculating the round trip delay between a receiver and its AP.

RMTP _ACK header contains an expected packet field and a bit-map vector. While sending a ACK packet, a receiver writes its current expected packet value and a bitmap of its receiver window to these fields.

4.2 RMTP Agents

In ns, agents are defined as data structures that represent endpoints where packets are constructed or consumed [39]. In our implementation, three agents, namely the Sender, Receiver and DR agents fulfill the whole requirement of RMTP. For example, a sender agent creates the data packets and multicast them to the whole group. When a receiver receives a data packet, it reads the data ID of the packet and updates its receiver window. If it detects a packet drop, it writes a “0” to the related bit of its bitmap vector and at the first ACK timer expiration, sends it to its AP in an ACK packet.

Agents are implemented in C++. Tcl linkage classes, written in C++ code are used to modify the parameters of these agents or starting/stopping some agents function from Tcl scripts written by users (Figure 4.1). Since RMTP is a timer based protocol, each agent has a timer class for scheduling agent-specific tasks. A timer may start by a Tcl command or an incoming packet may trig a timer. For example the sender timer for sending periodic SAP packets starts when the Tcl command “start” is executed, while the ACK timer of a receiver starts when the first data packet receives. Each timer has a timer interval which is bounded by an agent parameter or dynamically calculated during the simulation.

All agent parameters like timer intervals, packet size or number of packets can be set to a value through the Tcl scripts by the users. A default value for each parameter should be defined in the necessary ns2 files.

4.2.1. Sender Agent

An RMTP sender agent works at the sender node of the multicast group. A sender agent can be created through Tcl by the following code;

```
set agent_name [new Agent/RMTP/Sender]
```

And the parameters for this agent can be initiated in Tcl code. These are;

i. sap_interval_ ; time interval for sending SAP packets (in seconds)
ii. data_interval_ ; time interval for sending DATA packets (in seconds)
iii. num_of_packets_ ; total number of packets sent during a session.
iv. retx_interval_ ; time interval for sending retransmission packets (in seconds)
v. mult_thres_ ; threshold value for multicast retransmissions
vi. sender_win_size_ ; size of sender window

vii. dally_interval_ ; time for waiting before terminating a session (in seconds)
viii. packet_size_ ; packet size (in bytes)

Parameters "packet_size_" and "num_of_packets_" are defined for generating a packet traffic without using an additional ns traffic source. All other parameters are used as defined in protocol.

Two other classes support the RmtpSender class, these are RmtpSenderTimer class and RmtpSenderWindow class (Figure 4.1). RmtpSender has four objects of the RmtpSenderTimer class. These are;

i. sap_timer_ ; For sending periodic SAP packets. Starts with Tcl command "start"
ii. data_timer_ ; For sending burst of data packets at T_{send} interval. Starts with "start-data" command.
iii. retx_timer_ ; Timer for sending retransmissions. Starts with "start-data" command.
iv. dally_timer_ ; Timer for terminating the session. Starts after sending last packet.

Sender agent also has an object of RmtpSenderWindow class. This object keeps three main values of a sender window, namely swin_lb low bound of sender window (in other words, the minimum packet ID that has not been ACKed yet), send_next, the ID of the packet that will be send next and avail_win the number of packets that can be sent, when the data_timer _ expire (Figure 4.2).

A member function of RmtpSenderWindow is called while a new data packet is being generated by the agent and it returns the send_next value, if avail_win is greater than 0. Another member function is used for updating the values of window, i.e. the sender agent collects the incoming ACKs during a T_{send} interval and at the end of interval it calls this function with the minimum of the dropped packets (say n) that is reported by the receivers. Then the function increases the swin_lb and avail_win values until swin_lb reaches n.

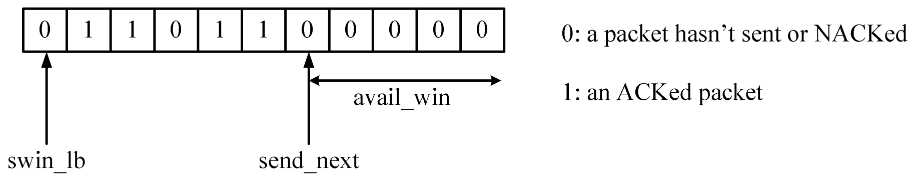

Figure 4.2 Sender window

The sender agent also has a map data structure for keeping the retransmission requests until the retx_timer expires. As it is illustrated in Figure 4.3, a retransmission map keeps a list of ns2 addresses for each data packet ID that is reported as lost by at least one receiver.

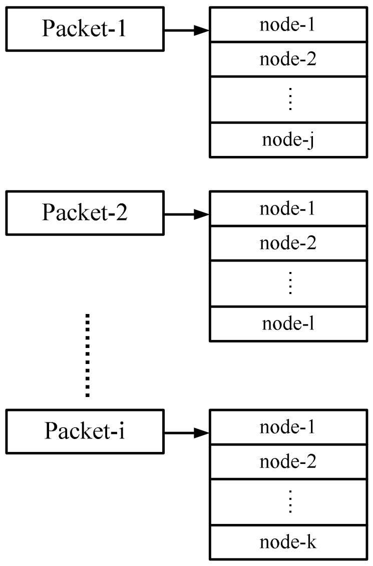

Figure 4.3 Retransmission map

For example, in Figure 4.3, all nodes from node-1 to node-k have sent a retransmission request for Packet-i.

By using these objects, a sender agent performs the following functions:

i. Sending periodic SAP packets to the whole multicast group.
ii. Sending RTT_ACK packet immediately after receiving a RTT packet.
iii. When data_timer_ expires, sending a burst of data packets as long as SenderWindow class returns a valid packet ID, in other words avail_win value is greater than zero.
iv. When an ACK packet is received, processing it and adding an entry to the retransmission map for the packet losses, and updating the sender window according to incoming ACKs.
v. When retx_timer expires, sending retransmission packet by unicast or multicast according to the number of receivers that have lost the packet.
vi. After sending the last packet, starting the dally_timer_ and terminating the session when the timer expires. If an ACK packet received while the timer is running, the dally_timer_ is reset to its initial value.

4.2.2. Receiver Agents

RMTP receiver agents work at the normal (i.e. not DR) receiver nodes. The Tcl code for creating a receiver agent is

```
set agent_name [new Agent/RMTP/Receiver]
```

and the parameters for the agent are

i. rtt_interval_ ; time interval for sending RTT packets (in seconds)
ii. receiver_win_size_ ; size of receiver window

There is no Tcl command for starting or stopping any function of a receiver agent. For a receiver a multicast session starts with the first incoming SAP and finishes when the last packet has been received and the receiver buffer is empty.

As sender agent the receiver agent also has two associative classes, RmtpReceiverTimer and RmtpReceiverWindow. There are two timer objects defined in a receiver agent. These are;

i. rtt_timer_ ; For sending periodic RTT packets. Starts with the first incoming SAP packet.
ii. ack_timer_ ; For sending ACK packets. Starts with the first incoming DATA packets. ACK intervals are calculated dynamically on the delay between receiver and its AP.

A receiver window is slightly more complicated than a sender window. It constitutes of an integer expected value (exp_pck) and a vector representing the receiver buffer. exp_pck is the minimum packet ID that has not been received yet, or detected as lost. When a packet whose ID is equal to exp_pac received, it is immediately sent to the application and exp_pck value is increased by “1”. If a packet with an ID greater than exp_pck is received, it means one or more packets has been lost. In this case, the packet is kept in the receiver buffer. Receiver window vector is implemented as a three state vector. Initially all values are set to “-1”, indicating the related packets have not reached yet. A “1” means the corresponding packet has arrived but it is being kept in the buffer due to loss of previous packets. Lost packets are shown by a “0” in the vector. For example, Figure 4.4 illustrates a receiver window where the receiver is waiting for the packet whose packet ID is 20. But it receives a packet with ID=23. Thus, it keeps this packet until packets 20, 21 and 22 arrive.

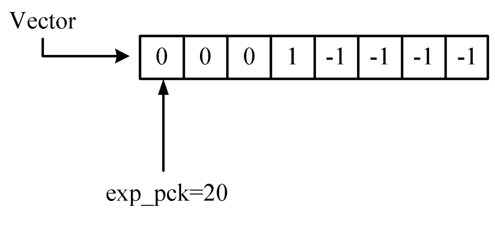

Figure 4.4 Receiver window

During a multicast session, a receiver agent performs the following tasks:

i. By using incoming SAP packets, selecting an AP (ACK Processor): The agent checks the TTL (Time To Live) field of the SAP packets. If it is greater than its stored TTL value, which means that the owner of that SAP packet is closer to the receiver, the receiver agent selects that node (it may be the sender or one of DRs) as its AP and stores the TTL value it retrieved from the SAP packet for comparing with the next SAP packets.
ii. After the first SAP packet received, the receiver agent starts to send periodic RTT packets to its AP in order to measure the delay between itself and the AP. The measured value is used to specify the interval for sending ACK packets.
iii. When a data packet is received, the agent modifies the receiver window. If the incoming packet is the expected one, the agent immediately sends the packet to the application and increases the expected packet value. In simulation, sending a packet to the application means deallocating the packet.
iv. When the ACK timer expires, the receiver agent sends a unicast ACK packet to its AP. An ACK packet contains the current expected packet value of the agent and a vector representing the receiver buffer.
v. When the receiver agent receives the last packet, it checks the receiver window. If the window is empty, that means the agent has received all packets. Then it stops sending periodic RTT and ACK packets.

4.2.3. DR Agent

A DR agent is a combination of a sender agent and a receiver agent with two trivial differences. The first one is that it has no sending new data function, i.e. it only sends retransmissions. And the second difference is that, before replying a retransmission request, it checks the receiver window to check whether it has the packet or not. If it hasn't received the packet yet, it stores the request for replying at the next transmission periods.

4.3 Operation steps of the RMTP Implementation

In order to simulate a multicast group which uses RMTP as reliable multicast protocol, the user should write a Tcl simulation script at first. This script contains the necessary Tcl codes for creating nodes and the links between them, and assigning the attributes for these nodes and links. More information for creating network topologies in Ns-2 can be found in [39]. After generating the whole topology, the user should create a RMTP sender agent and a group of RMTP receiver and DR agents, and attach them to the related nodes. The parameters of the agents can be set in this Tcl code. If not, the simulator uses the default values which are defined in Ns-2 libraries.

After all, the user should write two commands and the simulation times for the execution of these commands. The first one is "start" which starts the session by sending and receiving the control packets of the protocol, like SAP or RTT packets, and the second one is "start-data", which starts the data traffic.

In this section the operation of the whole implementation will be explained. This operation can be divided into the following steps;

1. Sender agent sends the first SAP packet:

When the simulator executes the command "start", the sender agent starts the session. It sets the sap_running flag to 1 and sends the first SAP packet to the whole group. A SAP packet contains no information, it is only used for its TTL value that is stored in the IP header. After sending the first SAP the agent reschedule the sap_timer by the sap interval parameter. At each expiration of the sap_timer, a SAP packet is sent and the timer is rescheduled again. This scheme provides periodic SAP packets as long as sap_running is equal to 1 (Figure 4.5).

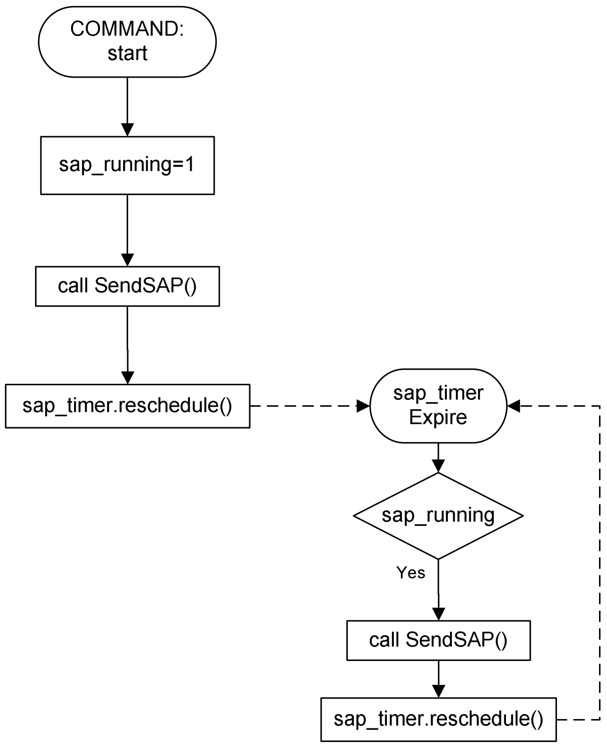

Figure 4.5 Starting the session by a user command

2. Receiver receives first SAP and starts periodic RTTs:

When a receiver agent receives a SAP packet, it checks whether the TTL value of the packet is greater than its stored *sap_* value. In constructor, a negative value is assigned to *sap_* . So if the SAP is the first one or it is sent by a closer AP (ACK Processor), TTL value would be greater than the *sap_* , thus the agent selects the owner of the SAP packet as its new AP. Then the receiver agent checks the rtt_running flag. If it is 0, that means the SAP packet is the first one. So the agent sets the flag to 1 and sends the first RTT packet. Then it reschedules the rtt_timer for sending periodic RTT packets (Figure 4.6). A RTT packet has a timestamp which indicates the send time of that packet and it is unicasted to the AP of the receiver.

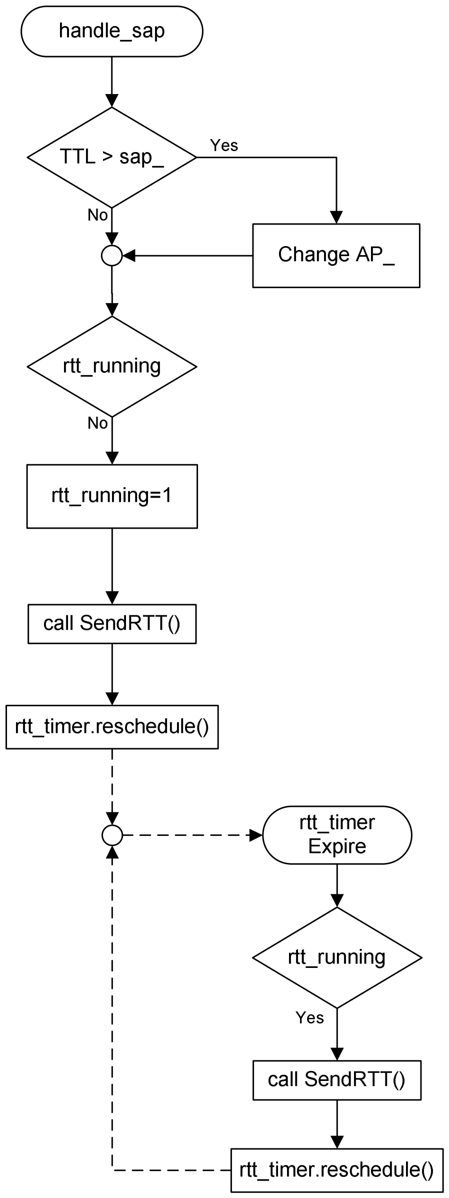

Figure 4.6 Start of a receiver agent with first incoming SAP

3. Sender agent replies RTTs with RTT_ACK packets:

When the sender agent receives a RTT packet, it immediately generates a RTT_ACK packet within the timestamp of the incoming RTT. Then the packet is sent to the owner of RTT by unicast.

When a receiver agent receives a RTT_ACK packet it easily calculates the round trip delay time between itself and its AP by subtracting the timestamp written in the packet from the current simulation time.

4. Sender agent starts sending DATA packet:

Only SAP, RTT and RTT_ACK packets are sent until the command "start_data" is executed by the simulator. When the command is executed, the sender agent sets the data_running and retx_running flags to 1 in order to start the periodic data and retransmission packet bursts. As seen in Figure 4.7, after setting data running flag, sender agent sends the first burst of data packets and reschedules the data_timer. When the data timer expires, the sender agent updates the sender window according to the ACKs received during last period, and sends a new burst of data packets unless the available window value of the sender window is zero. The SendBurst function sets a data_finished flag if it sends the last packet, DATA_EOF. As seen in Figure 4.7, simulator calls StopData function in order to terminate the session.

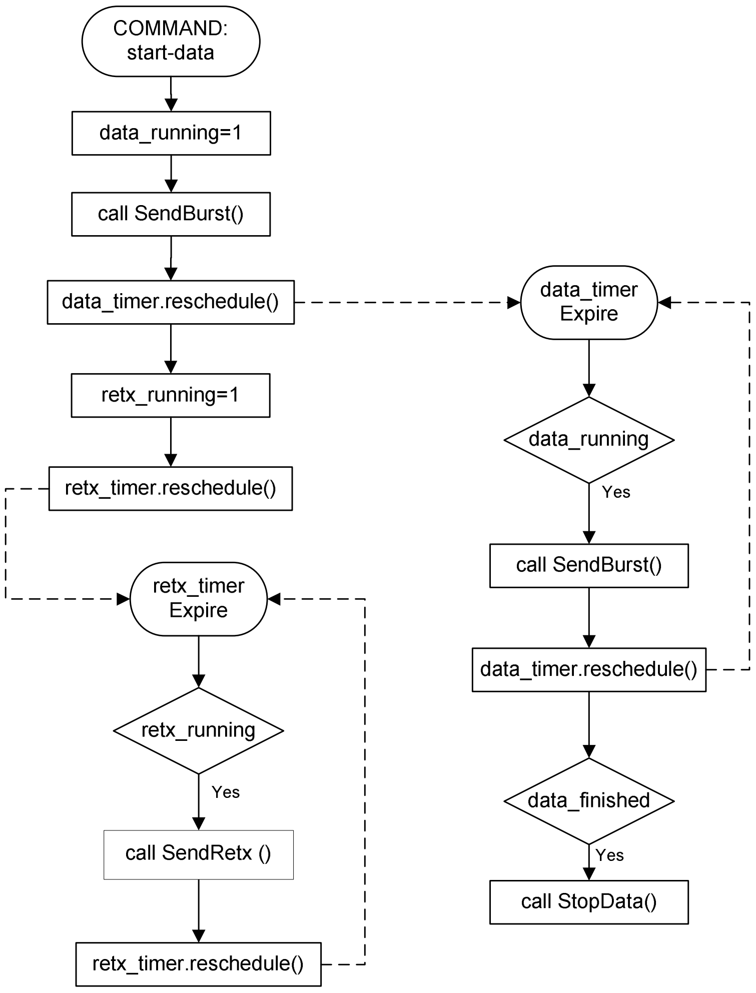

Figure 4.7 Starting DATA flow and sending retransmissions

When SendBurst function is called, it calls the Send member function of the sender window. This function returns a variable, *k*, which is going to be used as the session ID of the next data packet, if not equal to -1. A variable *k*, which is equals to -1, means the sender window is closed and no data packet can be sent until next data_timer expiration. If *k* is not equal to -1, the sender agent checks if it is last the packet and sends a DATA or DATA_EOF packet according to the result. As illustrated in Figure 4.8 if it is a DATA packet, agent calls the send function again for sending a new packet.

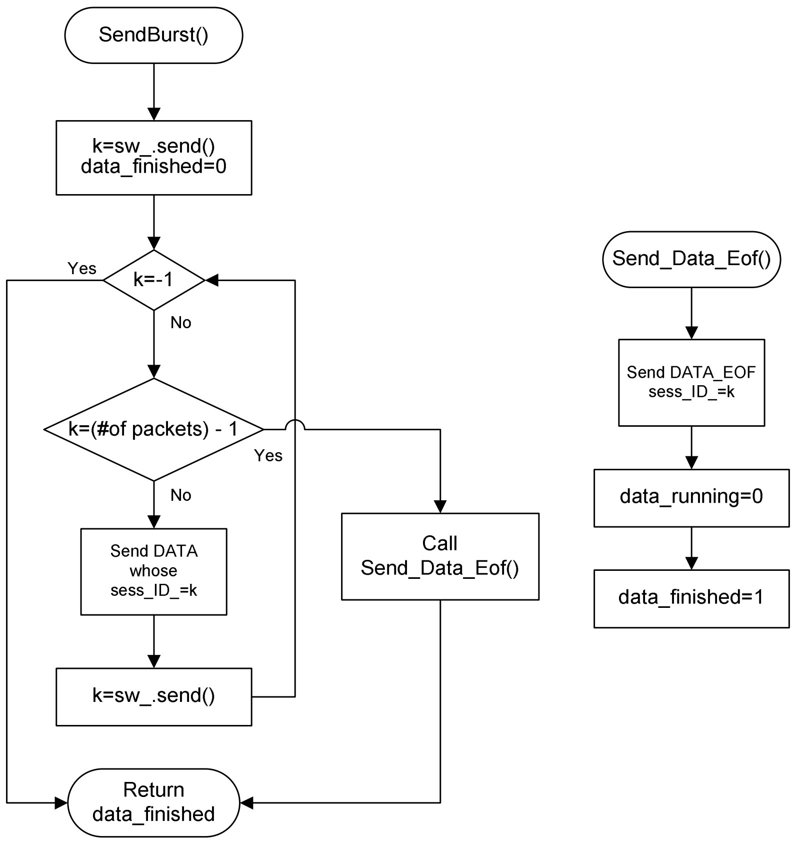

Figure 4.8 Sending a burst of DATA packets

5. Receiver agent receives DATA and starts sending ACK packets:

When a receiver agent receives a DATA packet, it checks the ack_running flag. If it is 0, that means the agent is not sending periodic ACKs currently. So it sets ack_running flag and schedules the ack_timer. After that, as seen in Figure 4.9, it calls Received member function of the receiver window. This function makes the necessary changes on the receiver window. When the ack_timer expires, the receiver agent sends an ACK packet and reschedules the timer (Figure 4.9).

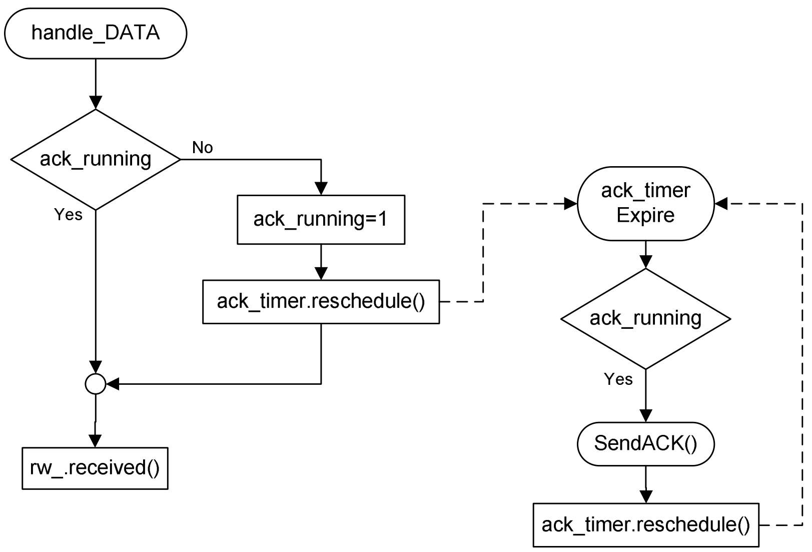

Figure 4.9 Handling DATA and sending periodic ACKs

When the sender agent receives an ACK packet, it adds the necessary entries to the retransmission map, retx_ , if the ACK packet reports any packet loss.

6. Sender agent sends retransmission as the retx_timer expires:

As it can seen in Figure 4.7, the first DATA packet starts the retransmission periods. When the retx_timer expires, the agent calls SendRetx function. This function scans the retx_ map and sends retransmissions by unicast or multicast, according to the number of receivers that required the corresponding retransmission.

7. Termination of a session:

In our implementation, a session may terminate in two ways. First, the user may write a “stop” command, to cancel all timers and stop data flows. But this is unreliable, since there may be some data retransmission packets that have not been sent yet.

In the second way, the method proposed by RMTP is used; After sending the last packet sender agent starts a timer, called dally_timer, with T_{dally} parameter. If an

ACK is received while the timer is running, it is reset. When the timer expires the sender agent stops all running timer, i.e. sap_timer and retx_timer. On the receiver side, when a receiver receives the last packet, DATA_EOF, it checks its receiver window. If it is empty, that means it has received all packets, then it stops sending periodic RTT and ACK packets. Otherwise, it keeps sending periodic packets until it receives all packets.

CHAPTER 5

SIMULATION STUDY

In the comparative part of this study, three selected protocols have been simulated in Network Simulator-2 under different test cases, for comparing some characteristics of these protocols. For this purpose, a single large network topology was created and the experiments designed for measuring different characteristics were performed on the same topology for each protocol. In order to obtain some quantitative results related to protocols, different evaluation metrics were defined.

In this section, details of the network topology, definition of the evaluation metrics, experiment design, results obtained from these experiments and the comments on these results are presented.

5.1 Network and application model

In network simulations, first step is to create a test topology on which all experiments will be performed. In literature, there are two different methods that are used for creating topologies with varying group sizes. The first one is to create the topology according to the number of multicast group members. In this scheme, all end nodes are group members. In the second method, an underlying network topology is created and than the group members are selected randomly from the end nodes of this network i.e. all nodes do not have to be a member of the multicast group. This method introduces the advantage of being able to analyze the behavior of a protocol against both a dense multicast group and a sparse one.

In this study, the latter method was used; a large network was created by using a topology generator tool, Tiers [40]. In Tiers, a network is created in a three level hierarchy, namely, a WAN, MANs under this WAN and LANs under each MAN. The tool supports only one WAN and the number of MANs, the number of LANs per MAN and the number of WAN, MAN and LAN nodes are specified by the user. In our topology, there are;

- a WAN consisting of 3 WAN nodes,
- 4 MANs, each having 2 MAN nodes,
- and 5 LANs per MAN, each having 5 LAN nodes (end nodes)

The whole topology contains 131 nodes but 100 of these are end nodes, so the members were selected within these 100 end nodes randomly. The network topology used throughout the experiments is presented in Figure 5.1.

While generating the topology, Tiers assigns link propagation delays according to the type of the link, i.e. a link between two LAN nodes (LAN-LAN link) or a link connecting a LAN to a MAN (MAN-LAN link) etc. (Figure 5.1) These delays were multiplied with a coefficient in order to obtain test cases with varying network diameter. Bandwidths of the links were specified as follows;

i.	WAN-WAN links;	34368 kbps	(E3 carrier)
ii.	WAN-MAN links;	8448 kbps	(E2 carrier)
iii.	MAN-MAN links;	8448 kbps	(E2 carrier)
iv.	MAN-LAN links;	2048 kbps	(E1 carrier)
v.	LAN-LAN links;	100 Mbps	(Ethernet)

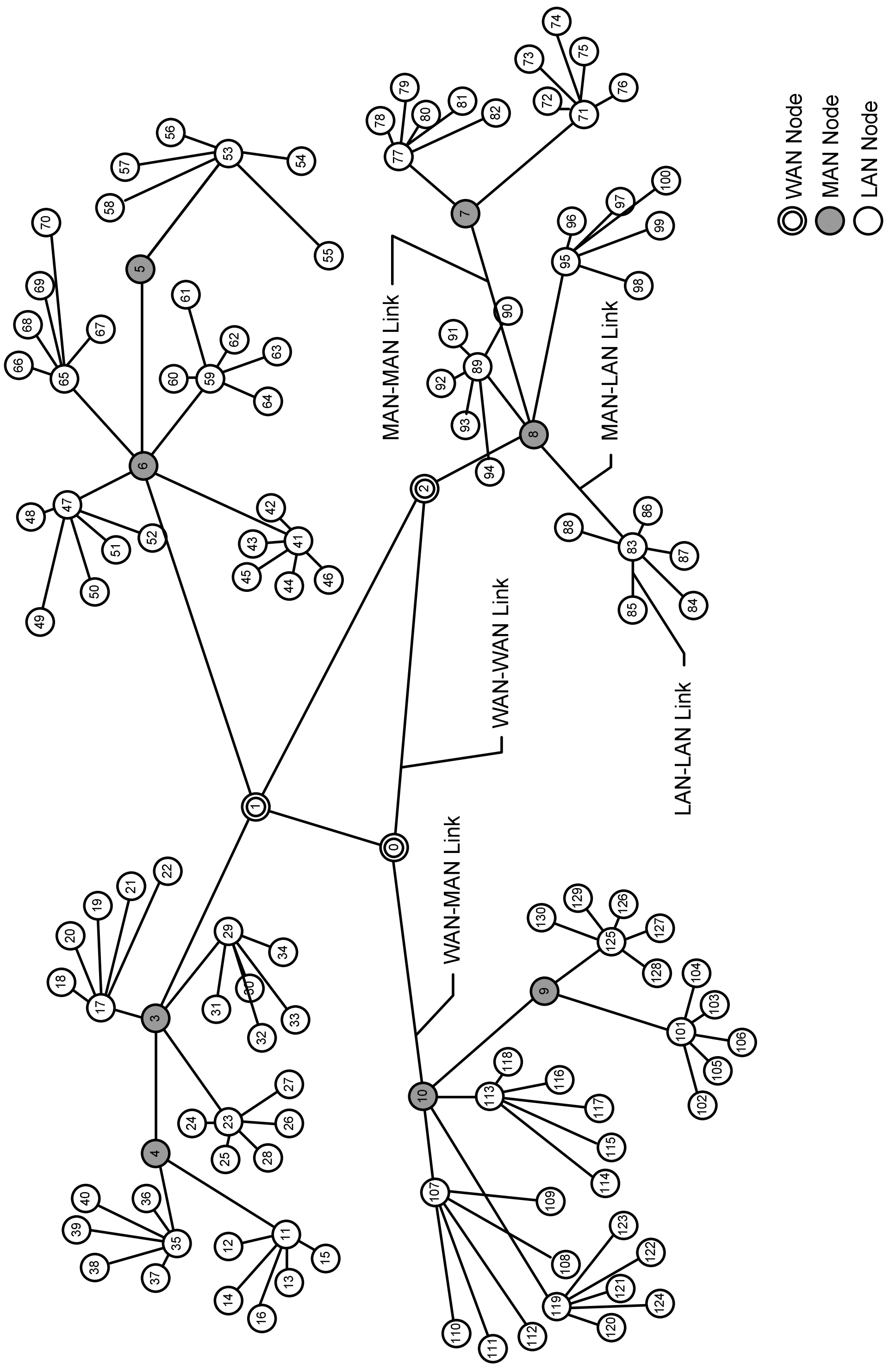

Figure 5.1 Underlying network topology

In real networks, most of the packet losses are due to the buffer overflows in routers. In order to simulate these events in a simulation, a background traffic, representing the normal traffic flow of the network, should be generated under the multicast traffic. But it is very difficult to generate a proper background traffic and it brings considerably large processing load to the simulation. Instead of this, in Ns-2, a user-defined loss rate can be assigned to each link. The simulator arbitrarily drops some packets passing over the link on an average rate that is defined by the user. In our simulations, all test cases are repeated under small and large loss rates, representing light and heavy background traffic, respectively. Furthermore, in order to observe the operation of a protocol on different network conditions, all evaluation metrics were measured with respect to varying loss rates.

Ns-2 supports three multicast route computation methods, named as Dense Mode (DM), Shared tree mode (ST) and Centralized multicast routing [39]. Dense mode multicast routing simulates the routing protocols DVMRP [8] or PIM-DM [41]. While creating a routing tree, prune messages are used in case of a node receives a packet for a group for which it has no downstream receiver. Similarly in Shared tree mode, prune messages travel between nodes until the routing tree is completely created. On the other hand, in Centralized multicast routing method, a centralized computation agent is used to compute the multicast tree and set up the forwarding states, and prune messages are not simulated [39]. Therefore when a simulation starts, it looks like all nodes know the routing tree and forward packets according to it, there are no join or prune messages. Since in this study, we deal with reliability, which is a transport layer issue, the creation of the routing tree has no significance. Thus Centralized multicast was used in all experiments.

Simulations were performed on one-to-many basis, i.e. a randomly selected sender sent a file of a predetermined size to a multicast group, which was also selected randomly. In SRM and PGM experiments, a CBR (Constant Bit Rate) traffic source was connected to the sender and it produced a packet traffic at 100 kbps. Since RMTP is window-based and its transmission rate is directly related to its protocol parameters like T_{send}, the RMTP sender agent was implemented in a way that it generates its own data traffic, without an external traffic source. But in all experiments, 100 packets were generated at sender nodes and transmitted to each group member, successfully. Each packet contains a data of 1000 bytes.

5.2 Evaluation metrics

Evaluation metrics are measurable parameters about the protocols that provide quantitative results from simulation executions allowing to comment on different characteristics. The metrics used in this study have been defined as follows:

Distribution Delay is the average time elapsed since a packet is sent from the sender until it has been correctly received by the whole group. The last group member that receives a packet specifies the distribution delay for that packet. If the packet has been dropped somewhere in the network, then distribution delay involves the time spent for recovery. From the point of view of the sender, distribution delay implies the minimum time for which the sender must keep the packet at memory before securely discarding it. Distribution delay is measured on a per packet basis. It is measured for each packet and then average of all is calculated.

Recovery Latency is the average time between a packet drop being detected by a receiver and its repair packet reaching the receiver. In other words, it is the time for which a receiver should wait in order to receive retransmission for a lost packet. It is measured for each packet loss during a session and the average is calculated at the end. If there is a back-off timer, which is required by the protocol as a delay before sending a retransmission request, it is included in the recovery latency.

Request overhead is the additional load on intermediate nodes (routers) generated by the protocol. In order to measure this parameter, the number of request packets processed by each router is counted and the average value is calculated over all routers who participate in the multicast distribution tree. The results are presented as the ratio of the average number of request packets to the number of original packets sent by the sender. Thus, it expresses the number of request packets required for sending a definite number of packets to a multicast group. In SRM and PGM, request packets are the NACK packets which are sent in case of a packet loss, whereas in RMTP, it is the ACK packets sent periodically by each receiver.

Repair overhead can be defined as similar to request overhead, except it is the load generated on each router by repair packets. The repair might be sent by the sender or another replier, like a DR in RMTP, and it might be unicast to the requester or multicast to the group.

5.3 Experiment Design

Before starting network simulations, different test cases should be generated, in order to determine the effects of some variable parameters of the network topology or the multicast group over the operation of the protocol. In this section the selected parameters and their range of variation will be presented.

The selected parameters used in this study are the following:

i. Group Size; It is the percentage of the number of group members to the total number of end nodes on the underlying topology. So that, group size gives an opinion about the density of the multicast, i.e. a low group size means a sparse group, and the density of the group increases as the group size increases.

ii. Group Diameter: In a network, the diameter can be defined as the distance between the end nodes. So it is directly related to the propagation delays of the links. In this study, delays were assigned by Tiers, while creating the topology. All values were multiplied by a coefficient, in order to obtain varying group diameters.

iii. Loss rate: As mentioned in Section 5.1, in order to simulate the background traffic on the network, a loss rate was assigned to each link. For observing the operation of the protocol in different network conditions, varying loss rates were used.

Table 5.1 gives the test cases generated by changing the parameters described above. In the first set of experiments, group size was increased from 10% to 90% while delay coefficient was kept constant at 1. Each case was repeated for a small and large loss rate, i.e. 1% and 5% respectively.

At the second set, group size was kept at 30% and delay coefficient was changed between 0.5 and 3. Again all experiments were performed twice, for small and large loss rates.

The last set of experiments were performed by altering the link loss rates for simulating different network conditions. In this case, group size and group diameter were kept constant.

Table 5.1 Test cases

	Network Parameters	Evaluation Metrics observed
Test Case 1	Group size = VARYING Network diameter = 1 Loss rate = 1%	Distribution delay, Recovery latency, Request overhead, Repair overhead
Test Case 2	Group size = VARYING Network diameter = 1 Loss rate = 5%	Distribution delay, Recovery latency, Request overhead, Repair overhead
Test Case 3	Group size = 30% Network diameter = VARYING Loss rate = 1%	Distribution delay, Recovery latency
Test Case 4	Group size = 30% Network diameter = VARYING Loss rate = 5%	Distribution delay, Recovery latency
Test Case 5	Group size = 30% Network diameter = 1 Loss rate = VARYING	Distribution delay, Recovery latency, Request overhead, Repair overhead

In order to increase the reliability of the simulation results, each experiment was repeated many times and the average of all was calculated at the end. The seed of the random generator of the simulator was selected in a way that it drops different packets at each repetition. The number of repetitions for each experiment was determined according to the confidence interval rule. This rule specifies the minimum number of repetitions which is required to obtain a simulation result that is at a predefined closeness to the real value.

In this study, all experiments were repeated until a simulation result, whose probability of being at ± 10% closeness of the real value is 95%, is obtained. After each simulation execution, the results were controlled according to confidence interval rule, and a decision is made on the termination of the corresponding experiment.

5.4 Simulation Results

In this section, results obtained from the simulations are presented in graphics showing the variation of the evaluation metrics against the network parameters. Numerical values of these graphics are given in Appendix A.

5.4.1 Distribution Delay

In Figure 5.2 - 5.6, the results obtained for distribution delay are presented. As it is seen from Figure 5.2. in case of small loss rate, increasing group size causes a smooth increase on distribution delay. It is expected, because as the number of members increases, the number of the members far from the sender will also increase and the number of packet drops will increase since these packets spread more on the links. In this case, results are very similar to each other. But in case of large loss rate, (Figure 5.3) some important changes on the operation of protocols are observed. First of all, characteristic of RMTP is very similar to small loss case except a rise which is due to the increase in the number of packet losses and retransmissions. But when the SRM graph is examined, it is seen that, in spite of rising group size, distribution delay is decreasing, in other words the protocol operates faster. This is the first important result, retrieved from the simulation work.

Since in SRM, each receiver could reply a retransmission request, it is reasonable that increasing the group size will decrease the retransmission time which will consequently decrease distribution delay. So SRM performs better at dense multicast groups. If we analyze Figure 5.2 and Figure 5.3 together, it is seen that the variation of SRM distribution delay is opposite, i.e. it is increasing in small loss case while decreasing in large loss rate. So that it can be said that, SRM operates better when the loss rate is higher. This is the result of using back-off timers before sending NACKs and repair packets. In case of high loss rate, the probability of receiving the same request or repair packet for which a receiver is running a back-off timer is higher, therefore recovering a loss packet would be faster. This case will be observed more obviously on recovery latency graphics.

Distribution delay for PGM is raising in both small and large loss rates, but rate of increase is considerably large at large loss. This is an expected result of using the original sender for all retransmission request. Since in this scheme, a retransmission packet travels a longer path between the sender and the requester, probability of dropping this packet somewhere on the path is much higher. This situation does not effect the operation at small losses but it is an important drawback when loss rate is high. As a result, PGM is badly effected from the changes in network condition.

Figure 5.4 and Figure 5.5 give the variation of distribution delay against network diameter for small and large loss rates, respectively. Operation of SRM and RMTP are similar in both cases but in high loss rate SRM again responds better than RMTP. PGM operation is unstable especially at high loss case.

Variation of a protocol against loss rate gives an opinion about the dependency of that protocol to the changes at network conditions. Figure 5.6 gives the operation at varying loss rates, or network conditions. For small loss rates, i.e. under 1%, results are very close to each other, which is supported by Figure 5.2 and Figure 5.4. But as the loss rate increases, SRM shows better performance than other two protocols. Furthermore, if the parts of the graphics between 2% and 4% of loss rates are considered, SRM graphics is almost horizontal. RMTP is also very close to a horizontal line but PGM distribution delay increases from 2.5 seconds to 3.1 seconds. That means PGM strictly depends on the network condition. Since other protocols use only a local part of the network while retransmitting a lost packet, they are less sensitive to network conditions than PGM.

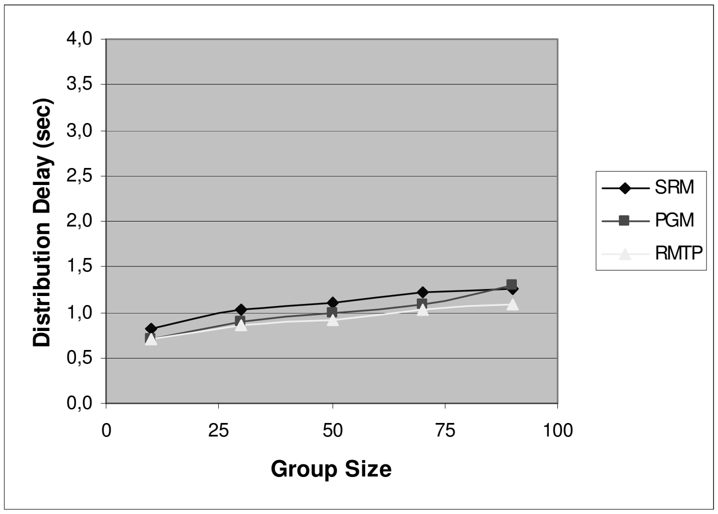

Figure 5.2 Distribution delay vs. Group size (for small loss rates)

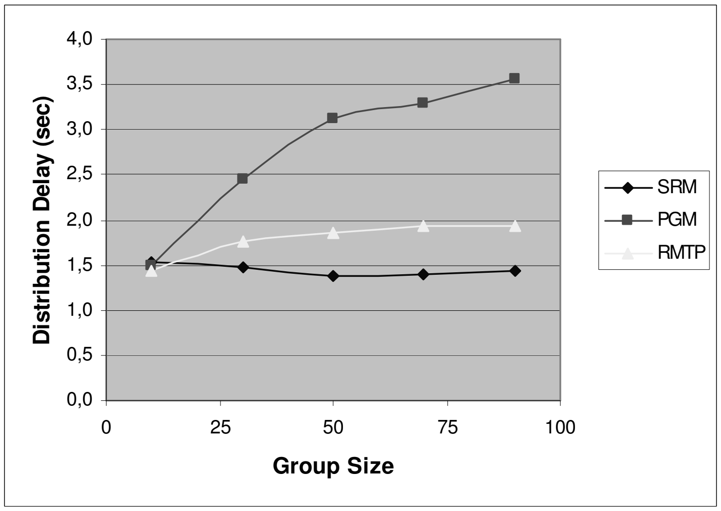

Figure 5.3 Distribution delay vs. Group size (for large loss rates)

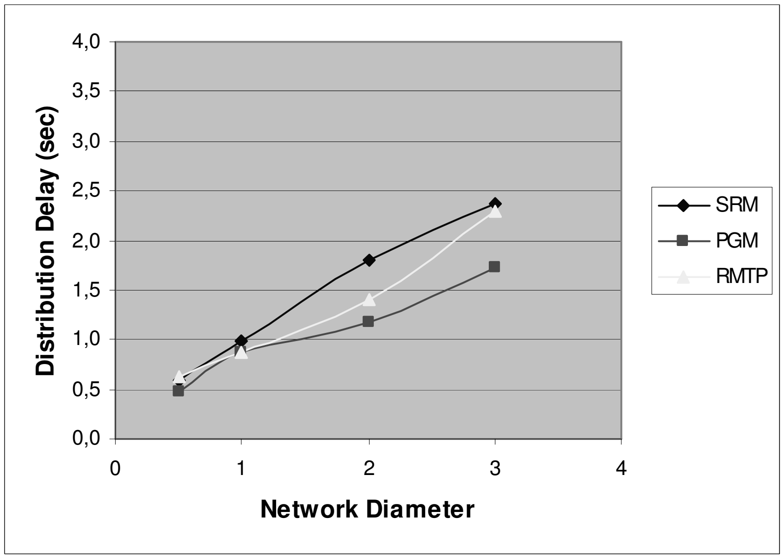

Figure 5.4 Distribution delay vs. Network diameter (for small loss rates)

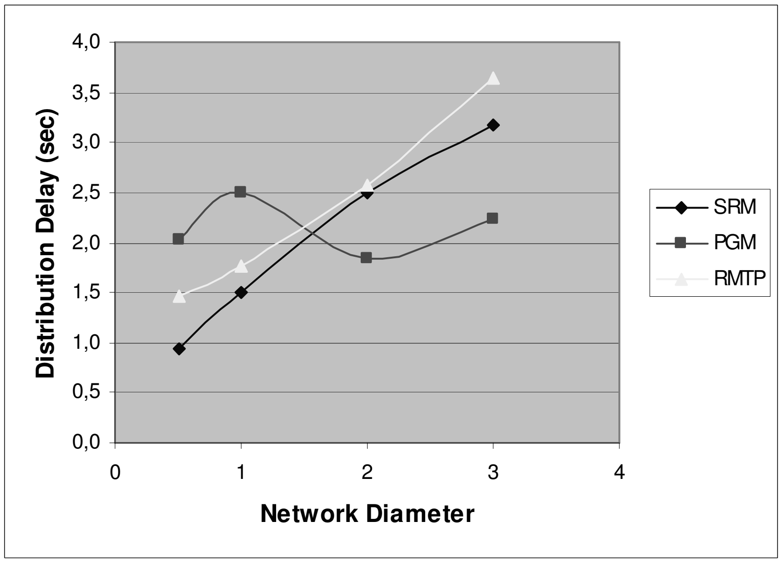

Figure 5.5 Distribution delay vs. Network diameter (for large loss rates)

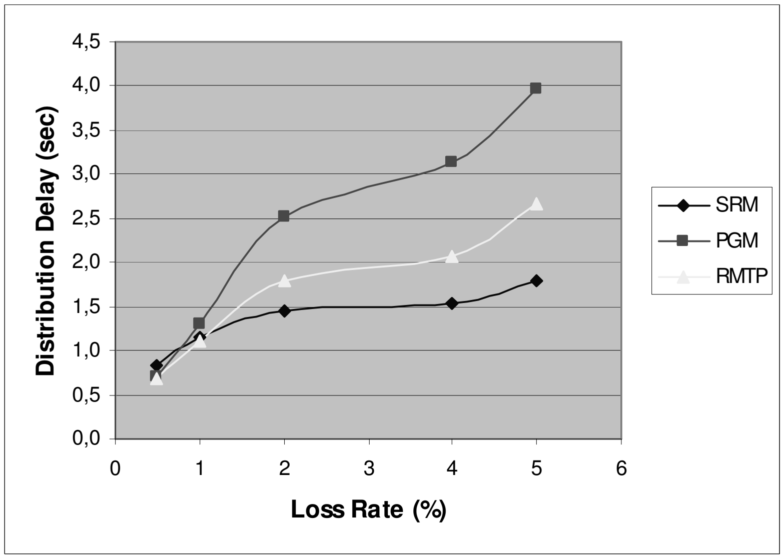

Figure 5.6 Distribution delay vs. Loss rate (Group size=30)

5.4.2 Recovery Latency

Graphics given in Figure 5.7 – 5.11, illustrate the variation of recovery latencies against different network parameters. As it is seen in Figure 5.7 and 5.8, recovery latency for SRM and RMTP decreases as the group size increases. Since these protocols use local retransmission, not being effected by growing group size is an expected result. Since RMTP uses multicast retransmission when the demands for a lost packet exceeds the $MCAST_{Thres}$ parameter, increasing group size causes more multicast retransmission, which consequently decreases the recovery time. But in both small and large loss rate cases, the characteristics of RMTP graphics are similar, which means recovery mechanism of the protocol is not affected from the network condition. The reason for the decrease at the SRM recovery times is also the local recovery mechanism. But SRM does not have local domains which have strict borders as in the case of RMTP.

The protocol adjusts random back-off timer values in order to supply the locality, as described in Section 3.4.2. But the mechanism requires dense multicast groups and high loss rates. As seen in Figure 5.8, SRM is the fastest protocol to recover a packet loss, when the group density is greater than 20%. If we consider the variation of PGM recovery latency against group size, it is almost doubled when network goes from small loss case to large loss case. That also supports the comment, given in previous subsection, which says increase on distribution delay arises from the long recovery times.

Figure 5.11 shows the affect of loss rate on recovery latency. Recovery mechanism of SRM and RMTP operate very stable in high loss rates, namely over 2%. But increasing loss rate raises the recovery time of PGM. That means PGM recovery mechanism is badly affected by changes in the network condition.

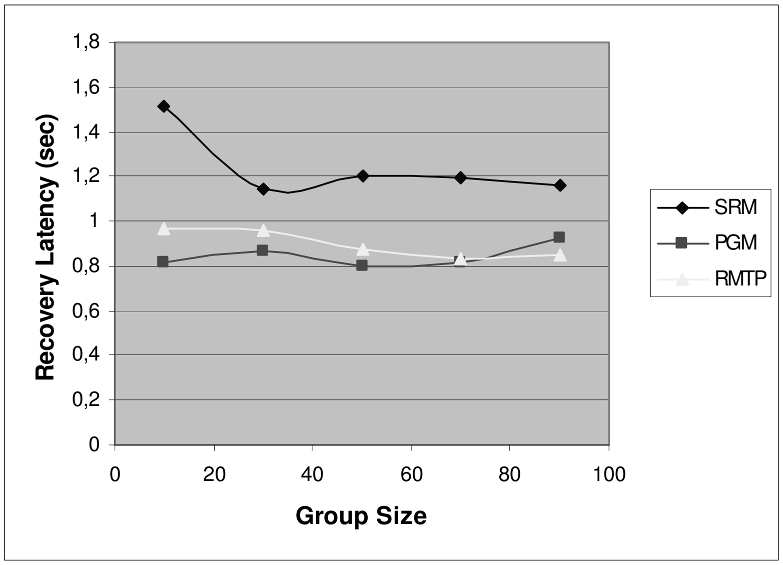

Figure 5.7 Recovery latency vs. Group size (for small loss rates)

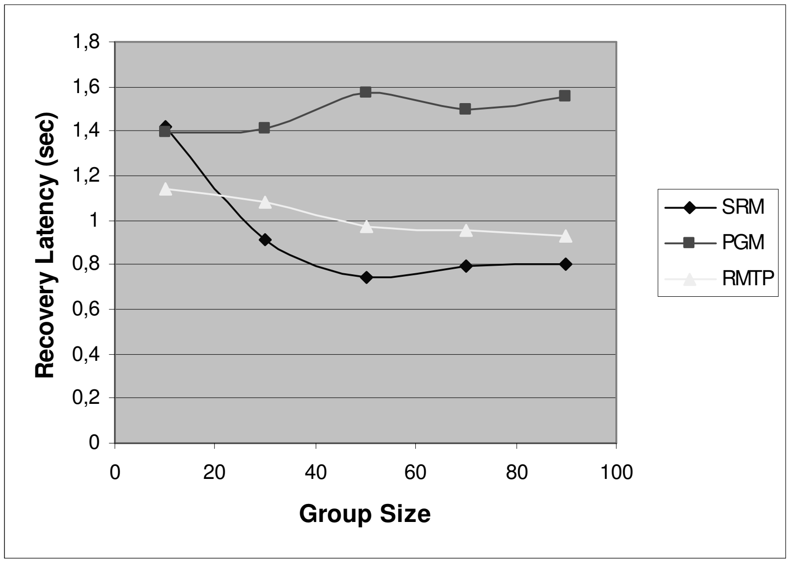

Figure 5.8 Recovery latency vs. Group size (for large loss rates)

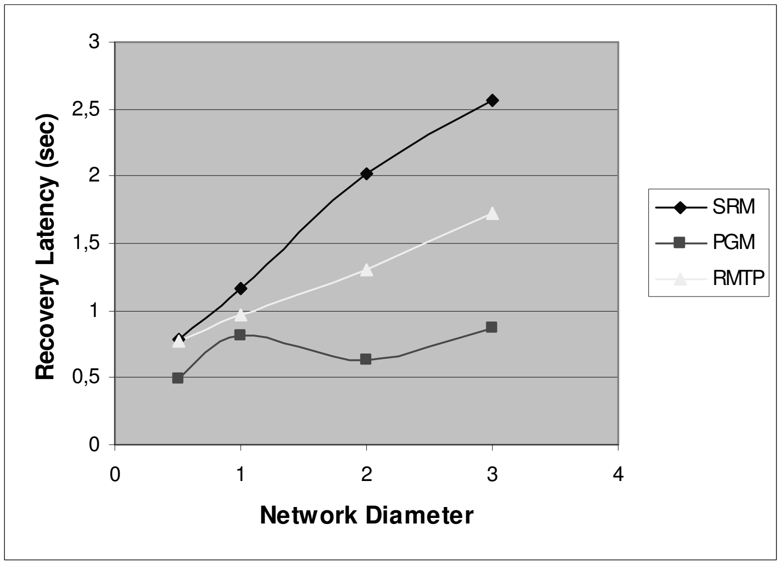

Figure 5.9 Recovery latency vs. Network diameter (for small loss rates)

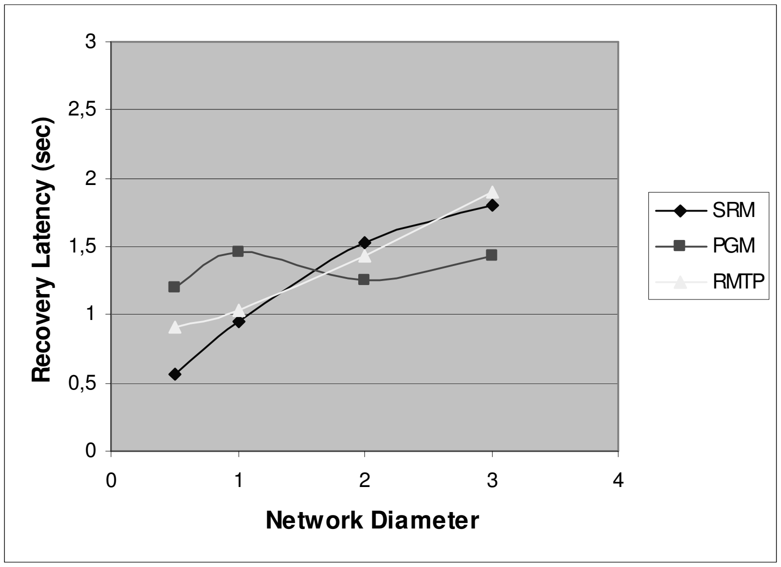

Figure 5.10 Recovery latency vs. Network diameter (for large loss rates)

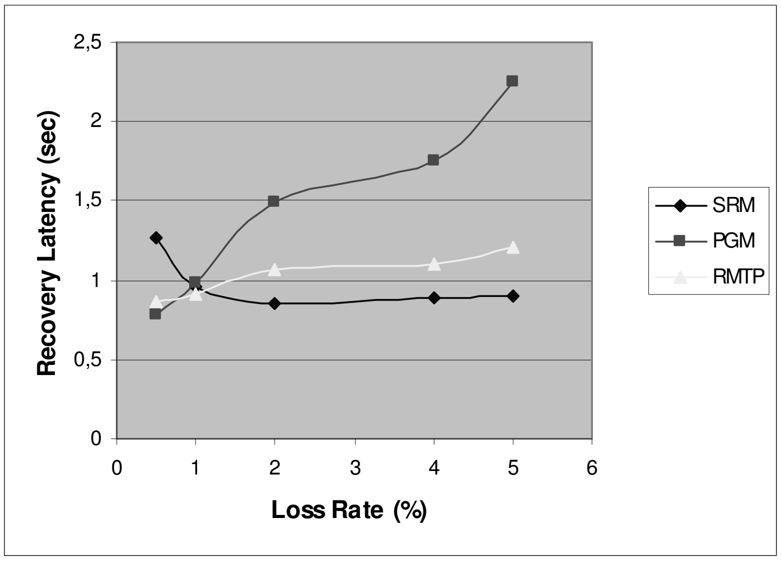

Figure 5.11 Recovery latency vs. Loss rate

5.4.3 Protocol Overheads

In Figure 5.12 – 5.17, the variations of the request and repair overheads generated by each protocol against group size and loss rate are given. Since the link delays do not effect the number of request or repair packets generated, variation against network diameter was not observed in these test cases.

In Figure 5.12 and 5.13, RMTP seems to generate more request overhead as the group size increases. The first reason for that is sending periodic ACKs for requesting lost packets, instead of sending NACKs only in case of a packet drop. Therefore request overhead of RMTP is higher than others in small loss rate case, and the difference gets higher as the group size increases (Figure 5.12). Another reason is not using a suppression mechanism for request packets like the other protocols. As a result of this, for increasing group sizes, RMTP request overhead increases faster than other protocols, especially in high loss case (Figure 5.13).

Since in a dense multicast group, it is likely to get a repair or request packet for a SRM receiver before its back-off timer expires and consequently its retransmission request would be discarded, SRM has a small request overhead at high group sizes, with respect to other protocols (Figure 5.13).

In case of PGM, since a request packet travels the whole network between the receiver and the sender, error probability for that request gets higher, as the link loss rates increase. Therefore especially in case of large loss rate (Figure 5.13), PGM generates more request overhead. For instance, request overhead for a group size of 10 is greater than 1, which means a router processes more request packets than the original data packets, even for a multicast group with 10 members. On the other hand, as it is seen in Figure 5.14, two protocols which use local retransmission do not suffer from increasing loss rate, as much as PGM. Furthermore, the behaviors of these protocols against loss rate are very similar to each other. Request overhead of both protocols remain under 1 until loss rate is equal to 4%, while PGM exceeds 1 at a loss rate of 1.5%.

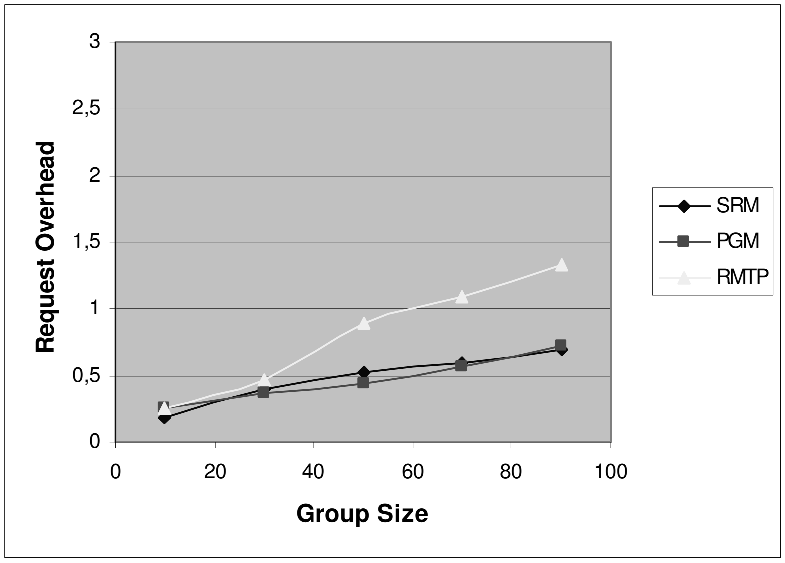

Figure 5.12 Request overhead vs. Group size (for small loss rates)

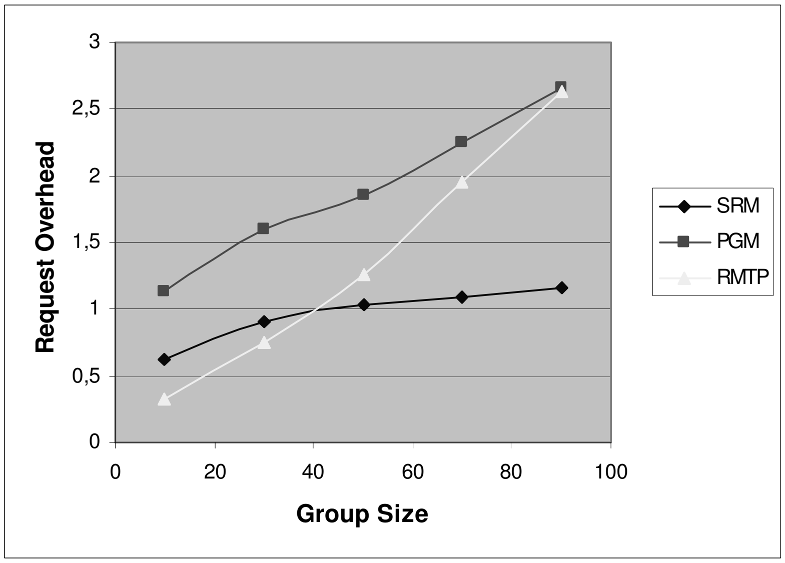

Figure 5.13 Request overhead vs. Group size (for large loss rates)

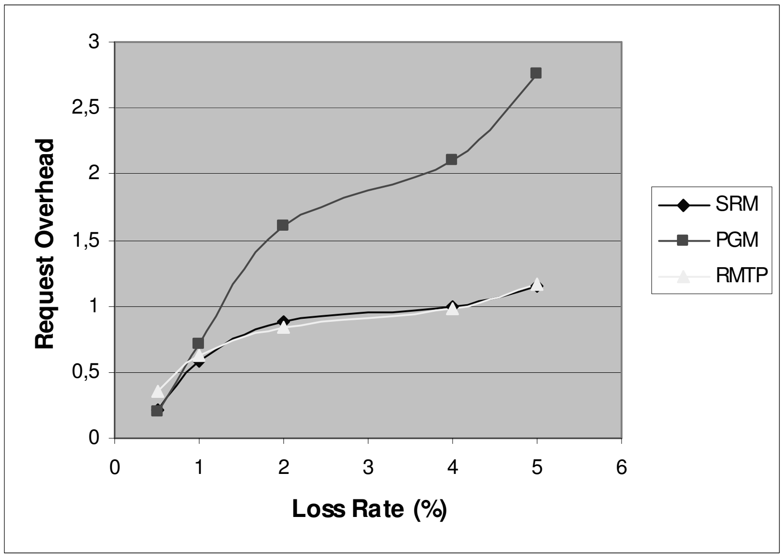

Figure 5.14 Request overhead vs. Loss rate

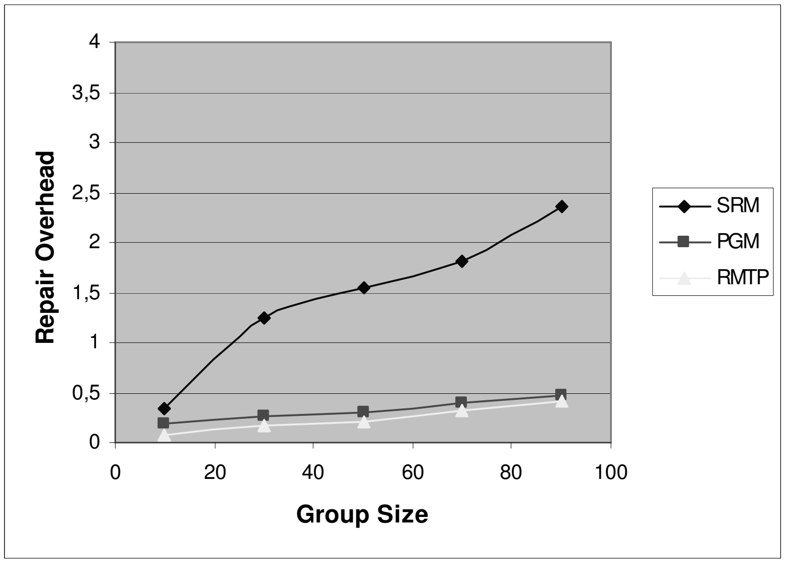

Figure 5.15 Repair overhead vs. Group size (for small loss rates)

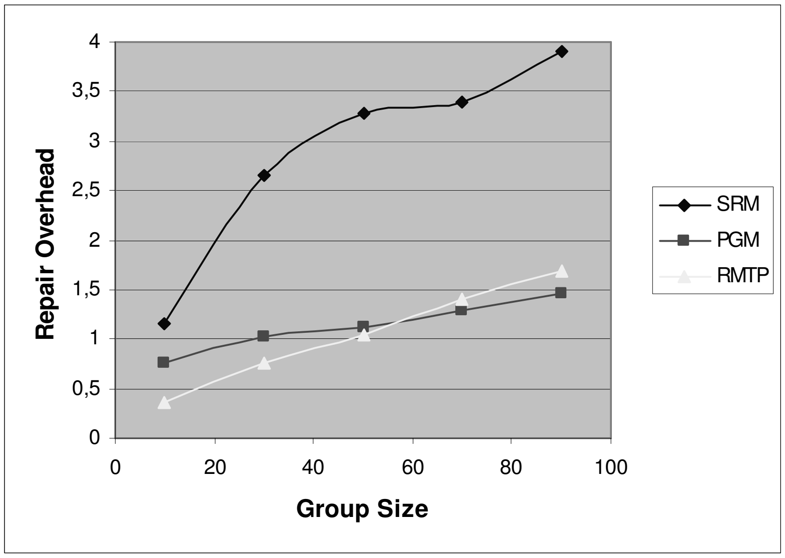

Figure 5.16 Repair overhead vs. Group size (for large loss rates)

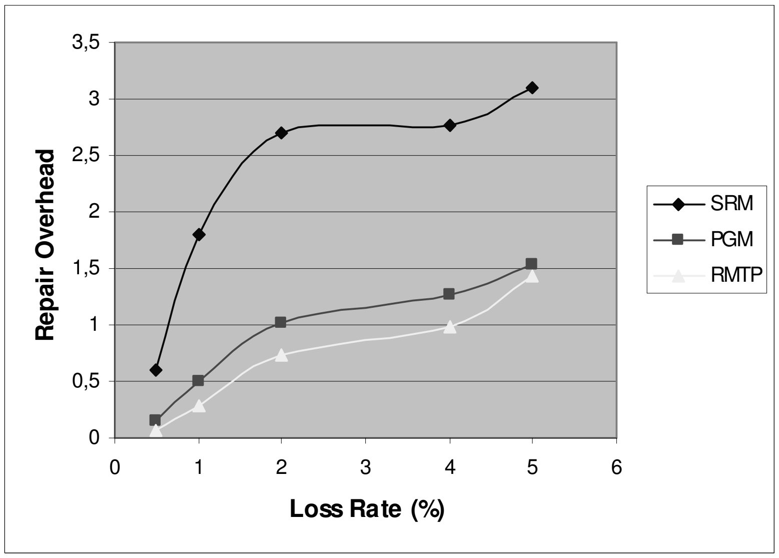

Figure 5.17 Repair overhead vs. Loss rate

From repair overhead graphics (Figure 5.15 and 5.16), it is seen that SRM generates a huge repair overhead even in small loss rates. This is a result of globally multicasting all repair packets. Multicast retransmission provides fast error recovery, because most of the receivers get retransmissions before expiration of their back-off timers. That can be recognized from the low request overhead of SRM. But this scheme costs a very high processing load on the routers. This is the trade-off of SRM for achieving fast error recovery.

From the figures, it can be said that RMTP and PGM generate an acceptable repair overhead in both small and large loss cases. In RMTP, sending retransmissions by unicast and using a local group when multicast transmission is required by the protocol keep the number of repair packets processed by each router at a low level. Consequently, the repair overhead is less than half of SRM's, which uses global multicast for all retransmissions. PGM also has a low repair overhead by taking the advantage of using router assistance. Repairs are sent by unicast and duplicated at routers if necessary, so repair traffic is kept low. But unicast retransmission causes high recovery times while generating less overhead.

CHAPTER 6

CONCLUSION

The overall evaluation of the results of simulation executions presented in previous chapter, gives some insight about the main issues of the reliable multicasting. In this section, these results will be summarized under the following distinctive titles;

Multicast retransmission: Sending retransmission packets by multicast transmission decreases the recovery time. SRM uses multicasting for all retransmissions and from simulation results it is seen to have a good recovery latency performance especially at high loss rates. Similarly, RMTP has a decreasing recovery latency. (RMTP is likely to send multicast retransmission in large groups.)

In case of multicast retransmission, some receivers get the lost packets before their back-off timers expire or while waiting for the next request sending period, decreasing the average recovery latency. This reduction is too obvious in SRM, such that it decreases the whole distribution delay. But while decreasing the recovery time, multicast retransmission introduces an extra repair overhead to the network. Especially SRM, which globally multicast all retransmissions, has a considerably high repair overhead. Instead of using global multicast, multicasting repairs to a local subgroup, such as RMTP, could keep the overhead at an acceptable rate.

Similarly, unicast retransmission cause a high overhead especially at the sender side of the network, i.e. one of the fundamental issues of reliable multicasting. PGM proposes to overcome this problem by using router assistance. This scheme is successful with respect to repair overhead, but since the retransmissions are not performed on a local-basis, recovery time is high. The effect of local retransmission will be investigated on the next item.

Local Retransmission: The term of local retransmission is used for the protocols in which not only the original sender but also a specialized receiver or any receivers who has the required data is able to reply a retransmission request. The most important advantage of this method is to keep the recovery times at a low level, as it is seen in SRM and RMTP simulations. A protocol that uses the sender for replying all retransmission requests, like PGM, would have a high recovery latency. But using a replier apart from the sender requires some additional protocol implementation efforts. For example, in SRM, each receiver should have the ability of sending a previously received packet when it is necessary. The same requirement is valid for a RMTP designated receiver, and furthermore selecting the locations of these DRs and running some extra features on them are necessary for a proper operation. These properties should be organized by a session manager. As a result, decreasing the recovery time by using local retransmission creates extra loads on protocol implementation and session management.

Effect of Network Conditions: In Ns-2 simulations, packet drops due to the network congestion are simulated by loss rates assigned to each link. Therefore variation of an evaluation metric against the loss rate gives an opinion about the response of the protocol to changing network conditions.

From simulation executions, it is seen that recovery time of the protocols using local retransmission, namely SRM and RMTP, is not effected from the changes in loss rate.

Out of the three protocols investigated in this study, SRM shows the best performance against variations of loss rate. Since in ideal operation of SRM the nearest receiver replies a retransmission request, the reply packet travels the minimum path when compared with other protocols and the effect of network condition is minimized.

In case of RMTP, each Designated Receiver specifies a local retransmission area and answers the retransmission request generated within the area. Hence RMTP also takes the advantages of using local retransmission. But since in this case each local retransmission area is specified by a Designated Receiver, they are larger than the local retransmission areas of SRM. As a result of this, effect of the network on RMTP recovery time is greater than it is in the SRM. But it is too little when compared with a protocol where the sender replies all retransmission requests. In PGM, for instance, recovery latency increases three times, as the average loss rate goes from 0.5% to 5%. So using local retransmission and decreasing the size of this local area decreases the dependency of a protocol on the network conditions.

Effect of Group Density: In this study, all test cases having different number of group members were created on the same underlying topology. Therefore varying the number of multicast group members caused a change on the density of multicast group, allowing to observe the effect of group density to the operation of the selected protocols. From the results obtained, a protocol that specifies the border of its local retransmission areas while establishing the session, e.g. RMTP, is independent of group density. On the other hand, since SRM does not define the local areas and obtains the locality by trying to answer a retransmission request by the nearest receiver, in sparse multicast groups the recovery latency was measured at the highest level and increasing the group size, i.e. the group density, caused a drop at the latency.

As a summary, using SRM, in a multicast session that requires reliable transmission, introduces fast error recovery, while creating a high protocol overhead. On the other hand, PGM is very slow at error recovery and badly affected by the changes at the network conditions. RMTP stays somewhere between these two protocols. Its error recovery times are quite low due to local retransmissions. Also, retransmitting repair packets to a restricted subgroup avoids generating large protocol overheads to the network. The only drawback of RMTP is the need of selecting and properly locating the DRs and the extra buffer requirement of these nodes.

At the end of this study, a combination of RMTP mechanism and the router support may be proposed, in order to overcome the reliability problem of multicast transmission in a scalable way. In such a scheme, the protocol may operate with less number of designated receivers, decreasing the extra buffer requirement and the difficulties of session management.

In this study, all simulations were performed with a single sender, i.e. on a one-to-many basis. As a future work, the response of the protocols in a many-to-many multicast session may be studied. The performance of the protocols with respect to some different evaluation metrics like processing load on sources, load on receivers or memory requirements of the protocols may be observed. Furthermore, CBR traffic sources were considered, during our study. The response of the protocols in case of different traffic types may also be studied.

REFERENCES

[1] Sportack, M. A., “IP Addressing Fundamentals”, Cisco Press, Indianapolis, 2003

[2] Li, V.O.K., Zhang, Z., “Internet multicast routing and transport control protocols”, Proceedings of the IEEE , Volume: 90 , Issue: 3, Pages:360 - 391, March 2002

[3] Tanenbaum, A. S., “Computer Networks”, Prentice Hall, September 2001

[4] Deering, S., “Host Extensions for IP Multicasting”, Network Working Group RFC 1112, August 1989

[5] Long, C., “IP Network Design”, McGraw-Hill, 2001

[6] Sahasrabuddhe, L.H., Mukherjee, B., “Multicast routing algorithms and protocols: A tutorial”, Network, IEEE, Volume: 14, Issue: 1, Pages: 90 - 102, Jan.-Feb. 2000

[7] Lin, J., Chang, R., “Comparison of the Internet multicast routing protocols”, Computer Communications, v 22, n 2, Pages: 144-155, 1999

[8] Waitzman, D., “Distance Vector Multicast Routing Protocol”, Network Working Group RFC 1075, November 1988

[9] Moy, J., “Multicast Extensions to OSPF”, Network Working Group RFC 1584, March 1994

[10] Ballardie, A., “Core Based Trees (CBT) Multicast Routing Architecture”, Network Working Group RFC2201, September 1997

[11] Kumar, S. et al., “The MASC/BGMP architecture for interdomain multicast routing”, Proc. ACM SIGCOMM, Vancouver, BC, Canada, Pages: 93-104, 1998

[12] Metz, C., “Reliable multicast: when many must absolutely positively receive it”, Internet Computing, IEEE, Volume: 2, Issue: 4, July-Aug. 1998, Pages: 9 – 13, 1998

[13] Miller, C.K., “Reliable multicast protocols: a practical view”, Local Computer Networks, 1997. Proceedings, 22nd Annual Conference on, 1997

[14] Mankin, A. et al., “IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols”, Network Working Group RFC 2357, June 1998

[15] Handley M. et al., “The Reliable Multicast Design Space for Bulk Data Transfer”, Network Working Group RFC 2887, August 2000

[16] Whetten, B., Taskale, G., “An overview of reliable multicast transport protocol II”, Network, IEEE, Volume: 14, Issue: 1, Pages: 37 - 47, Jan.-Feb 2000

[17] Lacher, M.S., Nonnenmacher, J., Biersack, E.W., “Performance comparison of centralized versus distributed error recovery for reliable multicast”, Networking, IEEE/ACM Transactions on, Volume: 8, Issue: 2, Pages: 224 – 238, April 2000

[18] Obraczka, K., “Multicast Transport Protocols: A survey and Taxonomy”, Communications Magazine, IEEE, Volume: 36, Issue: 1, Jan. 1998

[19] Levine, B.N., Garcia-Luna-Aceves, J.J., “A Comparison of Known Classes of Reliable Multicast Protocols”, International Conference on Network Protocols, Pages: 112 – 121, 1996

[20] Maihofer, C., Rothermel, K., Mantei, N., “A throughput analysis of reliable multicast transport protocols”, Computer Communications and Networks, 2000. Proceedings. Ninth International Conference on, Pages: 250 – 257, October 2000

[21] Appala, S.V.,Austen, J.R., “An evaluation of reliable multicast protocols”, Southeastcon '99. Proceedings. IEEE, Pages: 165-168, March 1999

[22] Whetten, B. et al., “Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer”, Network Working Group RFC 3048, January 2001

[23] Byers J.W., Luby M., Mitzenmacher M., Rege A., “A Digital Fountain Approach to Reliable Distribution of Bulk Data” Computer Communication Review 1998

[24] Paul, S., Lin, J., Sabnani, K., Bhattacharyya, S., “A Reliable Multicast Transport Protocol (RMTP)”, IEEE Journal on Selected Areas in Communications, Pages: 407 – 421, April 1997

[25] Yavatkar, R., Griffioen, J., Sudan, M., "A Reliable Dissemination Protocol for Interactive Collaborative Applications," Proc. ACM Multimedia '95, Pages 333 - 344, 1995.

[26] Floyd, S., Jacobson, V., Liu, C.G., McCanne, S., “A reliable multicast framework for light-weight sessions and application level framing,” IEEE/ACM Trans. Networking, vol. 5, pp. 784–803, Dec. 1997

[27] Speakman et al., “PGM Reliable Transport Protocol Specification”, Network Working Group RFC 3208, Dec. 2001

[28] Miller, K., Robertson, K., Tweedly, A., White, M. “StarBurst Multicast File Transfer Protocol (MFTP) Specification.” Work in progress, Internet Draft, draft-miller-mftp-spec-03.txt, April 1998.

[29] Papadopoulos, C., Parulkar, G., Varghese, G., “An error control scheme for large-scale multicast applications,” in Proc. IEEE INFOCOM, San Francisco, CA, Pages: 1188–1196, Mar.–Apr. 1998

[30] Costello, A. M., McCanne, S., “Search party: Using randomcast for reliable multicast with local recovery,” in Proc. IEEE INFOCOM, New York, Pages: 1256–1264, Mar. 1999

[31] Gao Y., Ge, Y., Hou, J.C., “RMCM: Reliable multicast for corebased multicast trees,” in Proc. IEEE Int. Conf. Network Protocols, Osaka, Japan, Pages: 83 – 94, Nov. 2000

[32] Bhattacharyya, S., Kurase, J., Towsley, D., “The Loss Path Multiplicity Problem in Multicast Congestion Control” Proc. IEEE INFOCOM, Pages: 856 – 863, Mar. 1999

[33] Pereira, J., Rodrigues, L., Oliveira, R. “Semantically Reliable Multicast: Definition, Implementation, and Performance Evaluation” IEEE Transactions On Computers, Vol. 52, No. 2, Pages:150 – 165, February 2003

[34] Ozkasap, O., Xiao, Z., Birman, K.P., “Scalability of Two Reliable Multicast Protocols” Technical Report, Department of Computer Science, Cornell University, New York, May 1999

[35] Hanle, C., Hofmann, M., “Performance comparison of Reliable Multicast Protocols using the Network Simulator ns-2”, 23rd Annual Conference on Local Computer Networks, 1998. LCN '98, Pages: 222 – 237, Oct. 1998

[36] Maihöfer, C., Rothermel K., “A Delay Analysis of Reliable Multicast Protocols”, IEEE International Conference on Multimedia and Expo, 2001. (ICME 2001), Pages: 92 – 95, Aug. 2001

[37] Maihöfer, C., Rothermel K., “A Delay Analysis of Tree-based Reliable Multicast Protocols” Tenth International Conference on Computer Communications and Networks, Pages: 274 – 281, Oct. 2001

[38] Gemmell, J., Montgomery, T., Speakman, T., Crowcroft, J., “The PGM Reliable Multicast Protocols” Network, IEEE, Volume: 17, Issue: 1, Pages: 16 – 22, Jan.-Feb. 2003

[39] University of Southern California Information Sciences Institute, The Network Simulator - ns-2, http://www.isi.edu/nsnam/ns/, January 2004

[40] Calvert, K.I., Doar, M.B., Zegura, E.W., “Modeling Internet Topology” Communications Magazine, IEEE, Volume: 35, Issue: 6, Pages: 160 - 163 June 1997

[41] Adams, A., Nicholas, J., Siadak, W., “Protocol independent multicast dense mode (PIM-DM): Protocol specification” Internet draft, draft-ietf-pim-dm-new-v2-01.txt, Feb. 2002.

APPENDIX A

NUMERICAL RESULTS OF SIMULATIONS

Table A.1 Simulation results for "Distribution Delay"

Group Size	Delay Coefficient	Loss Rate	Distribution Delay			
			SRM	PGM	RMTP	
10	1	small	0,82045	0,70509	0,70441	Graphic given in Figure 5.2
30	"	"	1,03315	0,89780	0,86520	
50	"	"	1,10590	0,98461	0,92141	
70	"	"	1,22760	1,07942	1,03459	
90	"	"	1,25755	1,29502	1,07993	
10	1	large	1,53669	1,49673	1,44009	Graphic given in Figure 5.3
30	"	"	1,48001	2,44977	1,76581	
50	"	"	1,37961	3,11802	1,85279	
70	"	"	1,39116	3,28483	1,92784	
90	"	"	1,42674	3,55178	1,93738	
30	0,5	small	0,59272	0,48140	0,62069	Graphic given in Figure 5.4
"	1	"	0,99297	0,87745	0,86588	
"	2	"	1,79260	1,16834	1,41102	
"	3	"	2,36665	1,72434	2,29886	
30	0,5	large	0,94046	2,03533	1,46229	Graphic given in Figure 5.5
"	1	"	1,50261	2,49647	1,76263	
"	2	"	2,48924	1,84209	2,57191	
"	3	"	3,18017	2,24347	3,65229	
30	1	0,5	0,82456	0,71000	0,68424	Graphic given in Figure 5.6
"	"	1	1,16169	1,29584	1,10392	
"	"	2	1,44703	2,51866	1,78715	
"	"	4	1,53114	3,12533	2,07129	
"	"	5	1,78295	3,96892	2,66876	

Table A.2 Simulation results for “Recovery Latency”

Group Size	Delay Coefficient	Loss Rate	Recovery Latency			
			SRM	PGM	RMTP	
10			1,51649	0,816508	0,965684	Graphic given in Figure 5.7
30	“	“	1,14706	0,870469	0,956497	
50	“	“	1,20004	0,797527	0,872792	
70	“	“	1,19782	0,814778	0,833078	
90	“	“	1,15731	0,926625	0,852444	
10	1	large	1,42039	1,391536	1,141568	Graphic given in Figure 5.8
30	“	“	0,909948	1,408138	1,081536	
50	“	“	0,746086	1,570705	0,971228	
70	“	“	0,792955	1,498271	0,953857	
90	“	“	0,801915	1,555407	0,931976	
30	0,5	small	0,782634	0,487545	0,768435	Graphic given in Figure 5.9
“	1	“	1,17024	0,811763	0,963149	
“	2	“	2,02321	0,628081	1,308198	
“	3	“	2,56688	0,874446	1,727937	
30	0,5	large	0,559879	1,203091	0,914317	Graphic given in Figure 5.10
“	1	“	0,952462	1,45463	1,033759	
“	2	“	1,52618	1,245698	1,424841	
“	3	“	1,80719	1,430183	1,905298	
30	1	0,5	1,27171	0,777217	0,867768	Graphic given in Figure 5.11
“	“	1	0,958407	0,979694	0,916683	
“	“	2	0,856448	1,487098	1,062336	
“	“	4	0,885987	1,758938	1,104222	
“	“	5	0,900743	2,255509	1,204661	

Table A.3 Simulation results for “Request Overhead”

Group Size	Delay Coefficient	Loss Rate	Request Overhead			
			SRM	PGM	RMTP	
10	1	small	0,180392	0,251523	0,251702	Graphic given in Figure 5.12
30	“	“	0,393386	0,367701	0,471496	
50	“	“	0,517599	0,434705	0,892522	
70	“	“	0,595226	0,572438	1,090108	
90	“	“	0,693548	0,717038	1,328635	
10	1	large	0,627132	1,130749	0,331925	Graphic given in Figure 5.13
30	“	“	0,908783	1,60596	0,748384	
50	“	“	1,032028	1,860176	1,257156	
70	“	“	1,085299	2,243812	1,953636	
90	“	“	1,160847	2,664076	2,635015	
30	1	0,5	0,215062	0,200328	0,362257	Graphic given in Figure 5.14
“	“	1	0,586667	0,714596	0,624646	
“	“	2	0,885344	1,608855	0,844411	
“	“	4	1,000159	2,105993	0,982862	
“	“	5	1,155397	2,763838	1,159562	

Table A.4 Simulation results for “Repair Overhead”

Group Size	Delay Coefficient	Loss Rate	Repair Overhead			
			SRM	PGM	RMTP	
10	1	small	0,33951	0,182634	0,075245	Graphic given in Figure 5.15
30	“	“	1,236614	0,260041	0,172194	
50	“	“	1,541505	0,301783	0,204516	
70	“	“	1,816839	0,388993	0,323118	
90	“	“	2,365887	0,472991	0,40866	
10	1	large	1,159559	0,761444	0,366791	Graphic given in Figure 5.16
30	“	“	2,656455	1,020774	0,754343	
50	“	“	3,274148	1,122434	1,03522	
70	“	“	3,390737	1,290704	1,397097	
90	“	“	3,899355	1,466892	1,691789	
30	1	0,5	0,594753	0,144559	0,073372	Graphic given in Figure 5.17
“	“	1	1,795556	0,494787	0,279125	
“	“	2	2,694656	1,02468	0,740976	
“	“	4	2,762275	1,260168	0,988889	
“	“	5	3,101164	1,535657	1,436936	