WAIT: Selective Loss Recovery For Multimedia Multicast

by

Pravin Mane

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

October 2000

APPROVED:

Prof. Mark Claypool, Advisor

Prof. Craig Wills, Reader

Prof. Micha Hofri, Department Head

Abstract

Recently the Internet has been increasingly used for multi-party applications like video-conferencing, video-on-demand and shared white-boards. Multicast extensions to IP to support multi-party applications are best effort, often resulting in packet loss within the network. Since some multicast applications can not tolerate packet loss, most of the existing reliable multicast schemes recover each and every lost packet. However, multimedia applications can tolerate a certain amount of packet loss and are sensitive to long recovery delays. We propose a new loss recovery technique that selectively repairs lost packets based upon the amount of packet loss and delay expected for the repair. Our technique sends a special WAIT message down the multicast tree in the event a loss is detected in order to reduce the number of retransmission requests. We also propose an efficient sender initiated multicast trace-route mechanism for determining the multicast topology and a mechanism to deliver the topology information to the multicast session participants. We evaluate our proposed technique using an event driven network simulator, comparing it with two popular reliable multicast protocols, SRM and PGM. We conclude that our proposed WAIT protocol can reduce the overhead on a multicast session as well as improve the average end-to-end latency of the session.

I dedicate this thesis to my wonderful family without whose love and support I would not have been who I am today.

The family I love the most...

the *MANE* family...

“ The most exciting phrase to hear in science, the one that heralds new discoveries, is not “Eureka!” (“I found it!”) but rather “hmm....that’s funny...”

–Isaac Asimov.

“ If a man will begin with certainties, he will end in doubts; but if he will be content to begin with doubts, he will end in certainties.”

-Francis Bacon (1561-1626)

“ May every young scientist remember and not fail to keep his eyes open for the possibility that an irritating failure of his apparatus to give consistent results may once or twice in a lifetime conceal an important discovery.”

- Patrick Blackett (British physicist, 1897-1974)

“ Keep on the lookout for novel ideas that others have used successfully. Your idea has to be original only in its adaptation to the problem you’re working on.”

- Thomas Edison (1847-1931).

“ Results! Why, man, I have gotten a lot of results. I know several thousand things that won’t work. ”

- Thomas Edison (1847-1931).

Acknowledgments

Prof. Mark Claypool, the advisor of this thesis, what can I say about such a great advisor? I can not find any words to show him my gratitude and appreciation for whatever he has done for me and this thesis. He is really a great mentor and a great advisor. I thank him for his patience that he has shown right from the beginning when I used to meet him with a lot of stupid ideas till the end of this thesis. His patience and showing me the proper direction when I seemed to be lost has proved to be very beneficial for me as well as my thesis. His support both as a mentor and a friend has made my stay at WPI a memorable experience.

I also thank my thesis reader, Prof. Craig Wills, for giving his precious time for reading my thesis and suggesting improvements in my thesis.

I am also grateful to the PEDS and PERFORM folks at WPI for providing me with necessary suggestions on my thesis.

I take this opportunity to thank all my friends at WPI who made my stay at WPI a memorable experience. The support, kindness and patience that they have shown for me for last few months is amazing. I specifically thank Badri and Ganga for providing me with valuable suggestions on my topic and Jae Chung for providing valuable tips on NS. I also thank Anshul for encouraging me to write this thesis in LaTeX. I also thank all my friends in Pune for their support and love for me.

Last but not the least, I thank Amar, Chidambar and Meenu for supporting me in my tough period.

I dedicate my thesis to all my family members, my friends and to Mark.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computer communication is becoming part of day-to-day life relatively very fast. Traditional computer communication modes were unicast (one-to-one) and broadcast (one-to-all).

In unicast communication, there is one sender and one receiver that take part in the communication process. The communication may be simplex communication wherein one participant is the sender of data and the other participant is receiver of the data or duplex communication wherein both the participants are active senders as well as active receivers. Traditional applications like FTP, e-mail, chat etc. make use of unicast communication.

In broadcast communication, there are one or more senders that send data to every other receiver in the network. The communication involves sending the data to every computer on the network. There are packet radio networks, satellite networks and bus local networks that use broadcast communication.

With the new emerging applications like video-conferencing, shared whiteboards, multi-user games etc. getting into the life of an average computer user, a new mode of communication called multicast has recently emerged. These applications must also

be supported by the existing computer communications infrastructure.These applications require that the data should be sent only to a set of participants (termed as a *group*), not to every other end points, by using effective means of communication. If unicast communication is used then the same data needs to be sent again and again to every other participant which is very inefficient. If broadcast communication is used then the data is delivered to those end points in a network that might not be interested in the data. Hence to fulfill the requirements where the communication is restricted to only a set of participants (a group), multicast communication emerged in the computer communications arena. Multicast facilitates the communication of only the group of receivers that are interested in the data. Multicast communication subsumes unicast and broadcast communications. Chapter 2 describes multicast communication in detail. Multicast applications can be classified into two types depending upon their requirements for being successful.

1. *Fully-Reliable Multicast Applications*: Applications like multiuser games and shared whiteboards require that they should receive each and every data packet sent by the sender. Examples are interactive simulations and software updates.

2. *Semi-Reliable Multicast Applications*: Applications like audio and video conferencing allow for some packet loss. However these applications have strict delay constraints as late arriving packets will be useless for the application due to the real time nature of the data. Examples are audio and video conferencing and Video-On-Demand services.

Apart from the applications mentioned above, multicast supports applications like updates to replicated databases, inter-process communication among different cooperating processes, etc.

The success of these various multicast applications depends upon efficient multicasting. For emerging highspeed networks, multicasting has already become an important

issue to tackle since it reduces the wastage of resources caused by transmitting unwanted data to some of the network components like routers or end hosts.

IP Multicast [1] uses User Datagram Protocol (UDP) as the transport layer protocol. UDP merely provides applications the ability to communicate using the unreliable connectionless packet delivery service. So, UDP packets can get lost in the network. This provides a *best-effort* delivery mechanism that can result in high packet loss in the presence of network congestion. A number of studies have been conducted for studying the loss characteristics on the MBONE for multicast sessions [2] [3]. Transmission Control Protocol (TCP) has been used for years as an effective means of using retransmissions to recover lost packets. However, a number of problems occur if a TCP-style sender-based approach is applied to a multicast distribution.

1. Since each data packet triggers an acknowledgment (ACK) from all receivers, the sender can be flooded with the ACK packets for the same data packet. This problem is popularly known as the ACK Implosion Effect [4].

2. If the sender is responsible for reliable delivery, it must continuously track the changing set of receivers and reception state for each receiver which is difficult to obtain as well as maintain.

Hence, the general principle in multicast loss recovery is to have receivers manage their own reliability requirements for reliable multicast [5].

Due to inherent nature of the data delivery mechanism where each packet gets duplicated on all the links leading to other participants in IP Multicast, there are two main characteristics of the effect of loss on the links in a multicast session.

Using the topology shown in Figure 1.1 for an illustration, losses in a multicast session can be divided into two classes:

1. *Global Loss*

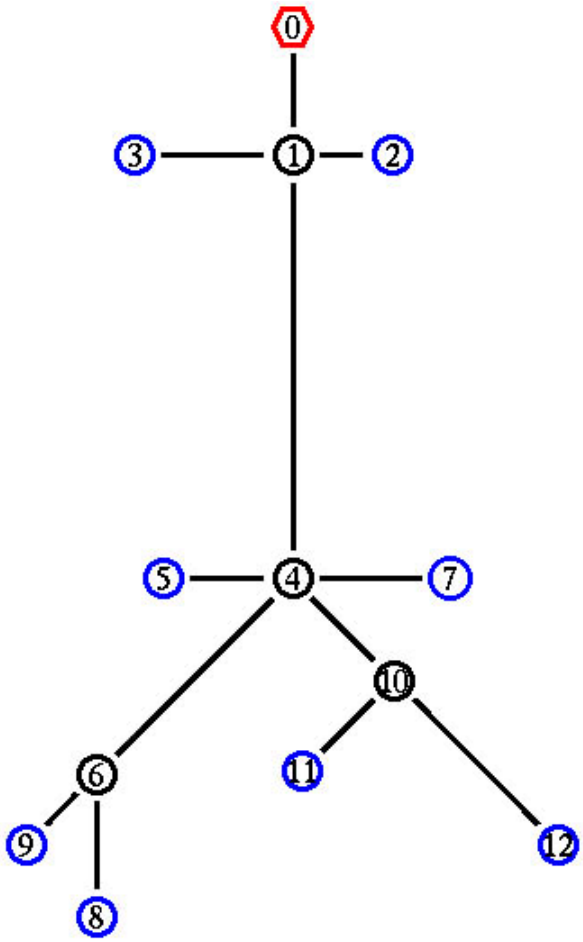

Figure 1.1: Multicast Tree

Loss occurs on the link 1-4, therefore all the receivers below this link 1-4 experience the same loss of packet i.e. receivers 5, 7, 8, 9 , 11 and 12 do not receive the packet lost on link 1-4. Loss experienced because of such links which are shared by different receivers can be termed as *Global Loss*.

2. *Local Loss*

   Loss occurs on the link 4-7, as a result only receiver 7 experiences the loss. Loss experienced because of such individual links can be termed as *Local Loss*.

In a multicast session, the sender can not determine whether the loss is *Global Loss*

or *Local Loss*. But receivers can group themselves to determine the type of loss as well as the link on which this loss has occurred. But in order to find the type of loss as well as location of loss in the multicast tree, receivers need to have some topology information.

The current problems facing receiver based reliability management in multicast are:

- *Implosion*: Simultaneous repair requests for one lost packet, characteristic of sender based loss recovery using retransmission. In the above example, when loss is experienced by all the receivers below the link 1-4 (*Global Loss*), each receiver can send a repair request to some other receiver or the sender.

- *Exposure*: Duplicate repair packets because of loss experienced by one receiver. In our scenario, if repair requests are delivered to receivers 2, 3 or the sender 0 then each receiver sends a repair packet for the loss and causing redundant repair packets being received by the receivers experiencing the loss.

- *Recovery Latency*: Time from when a receiver detects a loss until a repair for a lost packet is received. In case of loss on link 1-4 (*Global Loss*), the added time all the receivers 5, 7, 8, 9, 11, 12 takes to recover from the loss.

- *Unadaptability*: Inefficient repair in the presence of dynamic topology changes.

Multicast has some other concerns also. Scalability is an important issue in reliable multicasting. A multicast protocol is scalable if it can guarantee the desired quality of service to all the receivers (even in presence of thousands of receivers) in a multicast session. Another concern in multicast is of congestion control. A TCP friendly congestion control scheme which scales for a large set of receivers. The scheme should be fair to other TCP and multicast flows and be stable and responsive to network dynamics.

In recent years there have been a number of solutions proposed for reliable multicast that attempt to recover all the lost packets based on the assumption that communication

is not delay-sensitive but highly error-sensitive. This assumption however does not hold good for the real-time, high volume traffic of the continuous media streams. In traditional data communications, error-free transmission is achieved by adding error-detection information (usually a sequence number) to the transmitted streams and re-transmitting any lost packet. However, it is assumed that delay requirements of continuous media streams may not permit retransmissions. Since audio and video streams are less sensitive to error and can tolerate some packet loss, one approach to error control is to rely on increasing transmission overhead by including error correction rather than error-detection information. But this increases the bandwidth usage because of redundant information being sent across the network to achieve the desired quality.

There are various approaches being taken in the recent years for the reliable multicast problem. Some approaches are sender based and other approaches are receiver based. In a sender based approach, the sender is responsible for the reliable delivery of data to all the participants in a multicast session. In a receiver based approach, any receiver can be responsible for the reliable delivery of data to some other receiver.

Scalable Reliable Multicast (SRM) [6] is a reliable multicast protocol where the repair requests are multicasted to the entire group using a random backoff. The random backoff is used in order to avoid duplicate repair requests. The repair packets are also multicasted to the entire group using a random backoff again to suppress duplicate repairs. However, the use of back-offs increases the recovery latency and there may also be unnecessary repair requests and repair packets.

Light Multicast Session (LMS) [7] is another reliable multicast protocol which uses routers to aid in the loss recovery. Each router selects a replier link for retransmission and guides repair requests sent by the receivers to the replier link. Repair request from replier link is guided up the tree. The repair is then multicasted from the point where the loss has occurred. In some topologies, like a long chain topology, recovery latency is greatly

increased as the routers select replier links only from the downstream links.

Pragmatic General Multicast (PGM) [8] makes retransmission of the repair packet from either the source or some designated receivers. When a router receives a repair request (NACK) it forwards it on the upstream link towards the source and also sends a NACK confirmation on the link on which it received the NACK. It also creates a NACK state to suppress duplicate NACKs. The NACK flows all the way up the multicast tree towards the source and the repair packet flows only on the links from where the NACK(s) came for the repair packet. This indicates that the recovery latency for PGM is around one round trip time (RTT) to the sender of the multicast session. Since routers take active part in the losses experienced by the end hosts, there is an increased overhead on routers which also leads to the increased latency for other applications using the network.

Active Reliable Multicast (ARM) [9] is the most router assisted protocol. Here, routers at strategic locations perform best effort caching of the data packets to aid the receivers in the loss recovery. Routers drop duplicate requests and do partial multicasting of the repair packet.

All the above protocols are developed with applications like shared whiteboards and multiuser games requiring full loss recovery in mind. However, as we pointed out multimedia applications can tolerate certain amount of packet loss.

Structure Oriented Resilient Multicast (STORM)[10] is a multimedia multicast protocol in which group participants self organize themselves into a dynamically built distribution structure and use the structure to recover lost packets from adjacent nodes. STORM distributes NACKs and repair packets along the multicast structure. Each receiver selects a parent node depending upon loss experienced by the candidate parent node as well as its own playback buffer which determines the upper bound on recovery latency. The request and repair packets are sent using unicast mechanism because of which more number of requests and repairs might flow for the same lost packet. In case of Global loss, the over-

all request traffic increases as requests are sent to the receivers that are also experiencing the same loss. And repairs are sent to each receiver experiencing the loss using unicast reducing the number of repairs being sent in the tree experiencing the loss with the effect that some receivers do not recover from the lost packet.

All the above protocols either take help from routers to avoid the implosion and exposure problems in a multicast session or increase the recovery latency to avoid duplicates in the multicast session or add too much traffic in the network in terms of requests and repairs.

We propose a new technique called WAIT, which avoids the implosion and exposure problems by effective communication between the receivers. The participants in our loss recovery technique arrange themselves in groups depending upon the topology information as well as knowledge of the surrounding neighbors in a multicast session. By forming a group, session participants recover from the two types of losses (*Global Loss* and *Local Loss*) effectively. The decision about when to send the request packet and to which nearby receiver is decided dynamically based on delay and loss in order to reduce the recovery latency using the topology information. In case of *Global Loss*, a group head (selected depending upon its position in the local group in the multicast tree) sends a special WAIT packet to the other nearby receiver(s) informing them that the group head has also experienced the same loss as that of the other receivers and is taking the responsibility of the loss recovery process for them. Also, the receivers decide when to send the repair request depending upon the loss they experience (and depending upon their own quality requirement) as well as the expected recovery latency and whether they receive any WAIT packet from their replier receivers. We believe this technique effectively recovers from the loss of packets as well as maintains the desired quality (in terms of loss as well as delay) of the audio and/or video streams in a multicast session.

In order to form a group, receivers needs to have some topology information. For

the delivery of topology information we propose a new *multicast trace route mechanism* and *limited scope receiver information delivery mechanism*. In our proposed Sender Initiated Multicast Trace-Route mechanism, each multicast capable router adds its address in the trace-route packet (sent by the sender) before forwarding that packet to all the other links in the multicast tree for the session. To deliver information about the neighboring receivers, we propose the Limited Scope Receiver Information Sub-cast mechanism that uses TTL based scoping to send the information to only the neighbors of the receiver.

To evaluate our proposed WAIT protocol, we use an event driven network simulator called NS2 (Network Simulator - Version 2) [11]. NS2 can simulate a network of computers with mechanisms to simulate communication between the network nodes and carry out performance studies of various protocols as well as different router management techniques. NS2 supports various multicast routing protocols as well as the SRM [6] and PGM [8] protocols. We have carried out a performance comparison of WAIT with SRM and PGM in terms of *Recovery Percentage*, *Recovery Latency*, *percentage overhead on the receivers* and *multimedia application quality*.

The remainder of the thesis report is as follows: Chapter 2 presents the related work in the area of reliable as well as resilient (semi-reliable) multicast, and describes the requirements of multimedia applications. Chapter 3 presents our approach, the WAIT protocol, which can be used for reliable as well as semi-reliable data delivery in a multicast session. Chapter 4 presents the implementation details and the simulation setup for the evaluation of WAIT protocol. Chapter 5 deals with the evaluation of different WAIT protocol configurations as well as comparison of WAIT with SRM [6] and PGM [8]. Chapter 6 discusses future work and Chapter 7 concludes the thesis.

Chapter 2

Related Work

Multicast communication over the Internet is implemented using IP Multicast and MBone, as described in Section 2.1. IP Multicast and the MBone are used primarily for the multimedia applications like Video-On-Demand and audio and video conferencing. Section 2.2 describes multimedia or continuous media. Since IP Multicast is a ***best-effort*** service, there usually is some provision for recovering from the losses experienced by the receivers in a multicast session. Section 2.1 also describes loss over the MBone. With loss comes degraded quality of audio and video streams. A number of techniques have been proposed to counter the effects of the loss encountered by multicast streams. These loss recovery techniques can be broadly divided into

1. Sender based loss recovery (Section 2.3).

2. Receiver based loss recovery (Section 2.4).

2.1 MBone and IP Multicast

In 1992, the Internet Engineering Task Force (IETF) decided to build a virtual network that runs on top of existing Internet to fulfill the requirements of group-applications with

effective use of network resources. This gave rise to the today's MBone (Multicast Backbone) which works on the principles of IP Multicast[1]. The MBone consists of special multicast capable routers as well as traditional unicast routers.

IP Multicast provides an effective mechanism of disseminating data from a sender to a group of receivers. Instead of sending a separate copy of data to each individual receiver, the sender sends a single copy to all receivers, which reduces the network overhead in terms of router resources uses as well as provides for effective bandwidth utilization. To deliver the data to the authorized members, a multicast tree is setup in the network with the sender acting as the root of the tree and receivers acting as the leaves of the tree. The sender sends packets to a *group address* and receivers wishing to join the group simply inform a local designated router using IGMP protocol [12] and listen to the group address. Multicast routing protocols like CBT [13], DVMRP[14], PIM[15], MOSPF [16] can be used to deliver packets to the joining receivers. In the MBone, the most widely used routing protocol is Distance Vector Multicast routing protocol (DVMRP)[14].

The class D addressing scheme identifies a multicast packet. Class D addresses are identified by the top order bits set to be 1110 and the address range for Class D addresses are 224.0.0.0 to 239.255.255.255 (see Figure 2.1).

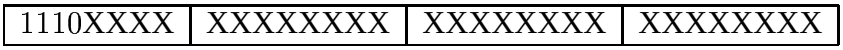

Figure 2.1: Multicast Address Format

This multicast address allocation is controlled by the Internet Assigned Numbers Authority (IANA) which generally does not assign individual IP multicast addresses to new application programs unless there is a strong technical justification. This leads to allocation of addresses dynamically. The most widely used method for Dynamic IP Multicast addressing is Session Directory program (SDR) [17].

Each multicast flow is identified by the sender address and destination group address.

When a multicast capable router receives a multicast packet, it checks for the interfaces inside its routing table which leads to the participants of a multicast session and sends the packet through all those interfaces only. If the router is not multicast capable, then the router encapsulates the multicast packet into an unicast packet and then sends the unicast packet to the next router in the path. This mechanism is called ***tunneling***. New commercial routers are becoming multicast capable eliminating the need and overhead of tunneling.

Since IP Multicast model is built upon the unreliable, best effort delivery using UDP as the transport layer protocol, losses are bound to occur within the network. The main cause of losses is router congestion. Previous studies has shown that 50% of the receivers have a mean loss rate of about 10% or lower [3]. 80% of the receivers experienced loss rate less than 20%. 80% of the receivers reported no loss during some interval of the day. During some interval of the day around 80% of the receivers experienced more than 20% loss. Around 80% of receivers reported some interval during the day when more than 20% loss was observed. About 30% of the receivers reported at least one interval where the loss rate was above 95%. They also showed that packet losses also occur in long bursts of 2-5 packets even though single packet losses dominate the overall loss pattern.

As the MBone is increasingly being used for audio and video conferencing, the amount of multicast traffic has increased tremendously. This demands for increase in the bandwidth as well as better multicast solutions to the problems described in Chapter 1. By proposing the WAIT protocol, we make an attempt to reduce the amount of traffic over the MBone by selectively asking for retransmission of the lost packets in the network. At the same time we try to provide the multimedia multicast applications the requested quality by recovering from the losses that occur over the MBone.

2.2 Continuous Media

With the increasing use of cost-effective audio and video hardware along with the availability of high speed networks providing high bandwidth necessary for continuous media communications, multimedia applications have been an effective mean for collaboration. Collaborative applications like video-conferencing, group-ware and computer-supported cooperative work can take advantage of efficient IP Multicast. However, communication using audio and video over the Internet faces some challenges.

1. *High volume of data*. CD-quality audio and HDTV-quality video require high rates of data transfer measured in Mb/s and Gb/s. Various compressions techniques developed so far are targeted to reduce the bandwidth utilization significantly.

2. *Interactivity*. There are strict delay requirements between the sender and the receiver to preserve the real time interactive nature of the multimedia applications. Past studies have shown that a certain amount of delay is tolerable by humans that can be between 40 and 600 milliseconds [18]. Along with this problem of bounded delay we have the problem of bounded delay variance called *jitter*. Jitter is normally dealt with by using buffering at the receiver and delaying the playback of received data. But this increases total delay experienced at the receiver with improvement in the playback quality.

Usually the audio and video streams can be transmitted by adding both audio and video data into one frame which greatly facilitates the inter-media synchronization as each frame is a synchronization point. But by transmitting each type of media independent of the other, we can treat each media independently, which allows us for media-specific treatment. This media-specific treatment provides for tolerating increased degradation of video quality than the audio quality degradation.

Multimedia applications require a lot of bandwidth and are also extremely delay sensitive. But loss of packets occur in the network, and receivers need to recover from the losses without incurring too much overhead. In this thesis, we are proposing a new approach for loss recovery during a multicast session, which selectively asks for retransmission of the lost packet(s) allowing the receiver to tolerate some loss depending upon the quality asked by the application. We also take into account the usefulness of the repair packet based on delay. If the repair packet is going to arrive late, our approach does not ask for retransmission, further reducing the overhead of packets.

2.3 Sender Based Loss Recovery

Sender based loss recovery can be split into passive channel encoding and active retransmission.

1. **Passive Channel Encoding**

   Passive channel encoding techniques are further divided into Interleaving and Forward error correction (FEC) techniques [19].

   In *Interleaving*, data units are resequenced before transmission and original adjacent units are separated by a guaranteed distance in the transmitted stream and are returned to their original order at the receiver. Interleaving disperses the effect of packet losses. A loss of single packet results in multiple small gaps in the reconstructed stream. In case of non-interleaved streams, a large gap occurs for each packet loss. A disadvantage of interleaving is increased latency, which limits the use of this technique only to the non-interactive application. A major advantage of interleaving is that it does not increase the bandwidth requirement of a stream.

   Forward Error Correction relies on addition of repair data to a stream. The repair data can be

(a) Independent of contents of the stream *or*

(b) Dependent upon the contents of the stream.

(a) **Media Independent FEC**

This approach uses codes to produce additional packets for transmission to aid the correction of losses. Each code takes a code word of k data packets and generates n-k additional check packets for transmission of n packets over the network.

The advantage of this technique is that since they are media independent, the operation of forward error correction does not depend upon contents of the packets and the repair is an exact replacement for a lost packet. Also the computation to derive error correction packets is relatively simple and small. But the disadvantage is that they incur additional delay, increased bandwidth and difficult decoder implementation.

(b) **Media Specific FEC**

In this approach each unit of data is transmitted in multiple packets. If a packet is lost then another packet containing the same unit is used to recover from the loss. Primary encoding (the first transmitted copy of the data) and secondary encoding, which is usually a low-bandwidth, lower quality encoding than the primary is used to achieve this. Use of media specific FEC incurs overhead in terms of packet size, which is variable and depends upon the encoding technique. The quality of repair varies with the overload. The advantage of media specific FEC is low-latency which is suitable for interactive applications where large end-to-end delay cannot be tolerated.

2. **Sender Based Retransmission**

Retransmission based schemes where in a loss of packet triggers the recovery process for the lost packet is another way to handle the packet loss. In a multicast session this type of mechanism should be handled in a very effective manner else there is overhead from repair requests and repairs in the multicast session. The ideal protocol for the loss recovery should send only one repair request to the sender or another nearby receiver who has received the packet. And the repair should be sent only to the receivers who have experienced the same loss. The recovery latency should also be taken into account for the real time multicast sessions where in late repair becomes useless because of the real time nature of the data.

As described above, one of the sender based approaches taken by the research community is that the sender has to take the responsibility of the reliable delivery of packets to the other participants in the multicast session. This increases the burden on sender as the sender has to keep track of the reception state of each receiver. Also in case of different receivers experiencing different losses, the sender will get flooded with the retransmission requests. The recovery latency for the lost packets can be effectively reduced if some nearby receiver helps the receiver experiencing loss to recover from the loss. We believe that this is very useful for the applications requiring real-time delivery of data and late data proves to be useless.

2.4 Receiver Based Loss Recovery

In recent years there has been a number of solutions / protocols proposed for reliable multicast that attempts to recover all lost packets. These protocols are called as *Reliable Multicast Protocols*.

Scalability is an important issue while designing a multicast transport protocol. Designing a multicast transport protocol that scales to large group sizes influences the design

of several functions like data propagation, reliability, repair request, feedback control, retransmission of lost data, flow and congestion control, locus of control, ordering, group management and target application. Some of the recently proposed protocols are:

1. Scalable Reliable Multicast (SRM) [6],
2. Light Multicast Services (LMS) [7],
3. Pragmatic General Multicast (PGM) [8],
4. Active Reliable Multicast (ARM) [9],
5. Tree Based Reliable Multicast Protocol (TRAM)[20].

All the above protocols are *Reliable* meaning that they care for each and every lost packet to be recovered. Since multimedia applications can tolerate certain amount of packet loss, there are some *Resilient Multicast Protocols* being developed such as Structure Oriented Reliable Multicast (STORM)[10].

2.4.1 Scalable Reliable Multicast (SRM)

Scalable Reliable Protocol (SRM)[6] is a prominent solution proposed with requirements of a shared white-board tool in mind. It works on the principles of Application Level Framing [21] and Light-Weight Sessions (LWS)[22]. ALF leaves as much of the functionality and flexibility as possible to the application. SRM uses multicast group concept of IP Multicast. Because of this concept, the data sources send data to the group's multicast address and receivers simply listens to that address, hence no knowledge of group membership or data sources is required. When a receiver detects a packet loss by detecting gap in the sequence number of the received packets the receiver multicasts a Negative Acknowledgment (NACK) to the entire group, closest receiver with the requested data re-

sponds with the requested packet. Repair packets are also multicasted to the entire group. This can result in NACK implosion as well as Exposure problem.

To avoid NACK implosion, SRM uses a randomized NACK transmission algorithm. When a receiver detects a loss, it does not transmit the NACK immediately but instead, waits for a random amount of time in hope that some other receiver might multicast a NACK for the same packet.

The back-off time is calculated by using the expression

back-off time = $D(C1 + C2 \times r)$

where :

D : estimate of one way delay between the source and the receiver.

$C1, C2$: Non-negative constants.

r : Uniformly distributed random number $[0, 1]$.

The $C1 \times D$ component of the above expression is called a *Deterministic delay* which is used to suppress duplicate NACKs from receivers located at different levels in the multicast tree.

The $C2 \times D \times r$ component of the above expression is called as *Random Delay* to suppress duplicate NACKs sent by receivers at the same level in the tree. The same back-off timer mechanism is used before sending the repair to avoid duplicate repairs. Since there are two back-off delays, the total recovery latency is greatly increased.

The scaling behavior of SRM depends upon topology of the underlying network as well as details of the timer algorithm. The values of $C1$ and $C2$ largely affect different requirements of different applications. Some applications require low recovery latency, but the expected latency is bounded by $(C1 + C2 \times f)$ where f is a function of network topology. Hence there is tradeoff between recovery latency and duplicate NACKs. The scaling behavior also depends upon an accurate estimate of the delay variable D. Also, as the number of receivers in the multicast session increases, the probability of any one

receiver experiencing a loss also increases, which has the effect that all other receivers will have to deal with NACKs and repairs. In the worst case, every packet sent might have to be retransmitted which effectively reduces the bandwidth consumption as well as induces more overhead on the receivers. When the group membership or network topology changes, the components for calculating back-off time have to be recalculated. This increases time to adapt to the best performance that can be offered by SRM. The performance and scalability of SRM suffers because of using global mechanisms to solve local problems.

To limit exposure, SRM with local recovery enhancements [23] uses TTL based scoping. However estimating appropriate TTL value is a unsolved problem.

2.4.2 Tree Based Schemes for loss Recovery

Tree based schemes (e.g. RMTP[24], TMTP[25], TRAM[20]) for loss recovery offer excellent scalability by arranging receivers in a tree hierarchy. Tree based schemes improve SRM's lack of local recovery and high recovery latency by arranging the receivers in a logical tree where the receivers will group themselves in local groups and elect a group head that will take the responsibility to cater to the losses experienced by its group members. But constructing a tree is a difficult problem since receivers do not have the topology information. Hence some tree-based schemes use a pre-configured hierarchy (RMTP) [24] and others use expensive mechanism to create and maintain tree (TMTP) [25]. But even these schemes do not shield bottleneck links from unnecessary request and repair traffic.

TRAM

TRAM [20] ensures reliability by using selective acknowledgment and scalability by adapting a hierarchical tree-based repair mechanism. The hierarchical tree not only elim-

inates implosion related problems but also enables localized multicast repairs. In TRAM, the receivers and sender interact with each other to dynamically form repair groups. These repair groups are linked together in a hierarchical manner to form a tree with the sender as the root of the tree. Every repair group has a receiver that functions as a group head. Except for the sender, every group head is a member of some other repair group. The group members report lost and successfully received data to the group head using the selective acknowledgment mechanism. To avoid overload, each repair head is responsible for only a limited number of receivers. The repair head caches every message sent by the sender and also provides repair service for the messages that are reported as lost by the members. If the group head does not have the data it asks for the data from the group head of which it is a member. Thus, there is a hierarchy of local groups to help with the recovery.

2.4.3 Network Support for Loss Recovery

All of the above schemes assumed that applications could not obtain help from the network routers, since routers are considered to be precious resources that have to forward packets at high rates and run several protocols. Also loss recovery is considered to be an end-to-end issue.

Because of the difficulties in Reliable Multicast, researchers are now examining the possibility of network involvement. The topology information available at routers can be used to avoid the implosion and exposure problems by restricting the traffic of NACKs and repair packets only to the part of the tree (called as loss sub-tree) that is experiencing the loss since loss on one link results in all receivers down that link experiencing the same loss. In last few years there has been new schemes proposed that use varying degrees of network support. Some of these includes Light Multicast Services (LMS) [7], Pragmatic General Multicast (PGM) [8], Active Reliable Multicast (ARM) [9].

Light Multicast Sessions (LMS)

LMS [7] allows any receiver to retransmit repair packets. There are three basic principles on which LMS works:

- Each router selects a replier link for retransmission.
- Routers guide NACKs to appropriate receivers, and
- Routers multicast repairs only from the root of loss sub-tree.

Each router selects its replier link based on the loss experienced by the receiver attached to that link. Upon detecting a loss, receivers will multicast a NACK with a special IP option so that routers can recognize such NACKs. Routers forward NACKs to the replier interface except NACKs from the replier interface are forwarded to the upstream router. The router that forwards NACKs to a replier link (the replier link above the point of loss which has the requested data packet) is called the *turning point* that is considered to be the root of the loss sub-tree. The replier then unicasts the repair packet to the turning point which in turn multicasts the packet downstream. However, since LMS assumes that the replier link represents the loss sub-tree, if only the replier link experiences a loss then the entire downstream tree will receive duplicates which will lead to the exposure problem.

Another problem with LMS is that it selects the repliers only from the downstream links that will increase recovery latency in some topologies like long chain topologies.

Also LMS can over react to NACKs for packets that have just been retransmitted since such NACKs can represent a late arrival of the NACK, and this might lead to exposure since the replier will retransmit the repair again.

Pragmatic General Multicast (PGM)

PGM [8] is the most heavyweight of the router-assisted protocols so far. PGM requires

that retransmission of repair packets should be done from the source or from some Designated Local Re-transmitters (DLRs). A DLR must lie directly on the path from the source. After detecting loss, a receiver selects a random back-off interval and then unicasts a NACK to the upstream PGM router. When a PGM router receives a NACK it multicasts a NACK Confirmation (NCF) on the link from where it received the NACK and unicasts the NACK to the upstream router. At the same time the router creates a NACK state for the corresponding sequence number. NCFs are used to acknowledge reception of the NACK. Routers suppress similar NACKs from other receivers by checking the NACK states. After the source receives a NACK, it retransmits with repair (RDATA) packet. Each router forwards RDATA on links from which the router received NACKs for the packet. After the router forwards the RDATA it discards the corresponding NACK state. This leads to the *Dangling NACK State problem* i.e. the NACK state is not discarded at routers until NACK state at routers expires when the RDATA packet is lost. Hence, if a receiver did not receive RDATA and times out and sends a NACK then the routers will incorrectly assume that there is recovery process going on upstream the tree and it will drop the NACK. This can lead to multiple retransmissions of RDATA packets. Also in some topologies where the distance between receivers is large, a receiver close to the loss may send a NACK and trigger a retransmission before NACKs from distant downstream receivers have a chance to establish NACK state all the way to the source. Since the NACK state is wiped out by the RDATA, a NACK arriving at a router after a RDATA has passed will reestablish the NACK state all the way to the source this will lead to repeated retransmission and exposure problem.

Active Reliable Multicast (ARM)

ARM [9] is also another reliable multicast protocol based on the principles of Active Networking technology. ARM utilizes router based error recovery to reduce end-to-end wide

area latency and to distribute the load of retransmissions. ARM uses intermediate routers to protect the sender and network bandwidth from unnecessary feedback and repair traffic. Intermediate routers perform following functions:

1. *Data caching for local retransmissions*. Routers at strategic points make 'best effort' caching of data for possible retransmissions. The time to cache data is approximated as a function of the inter-packet sending rate and the max RTT (Round Trip Time) between the sender and the "farthest" receiver downstream.

2. *NACK fusion and suppression*. Routers control implosion by dropping duplicate NACKs and forwarding only one NACK upstream towards the source. When a router receives a NACK, it retransmits the repair if it is in its cache, else it sends the NACK towards the sender and makes an entry into a subscription bitmap about on which link that NACK came. Routers control implosion by dropping duplicate NACKs by looking at the entries in the subscription bitmap and sending only one NACK upstream.

3. *Partial Multicasting for scoped retransmissions*. Routers perform multicasting of retransmitted packets so that they are delivered only to receivers that previously requested them. Retransmission by routers is done using partial multicasting to the receivers who requested that packet by using a subscription bitmap. Routers also maintain a repair record that indicates on which links the router has already forwarded the repair packet. This repair record is used to suppress NACKs sent by the receivers before they receive the repair packet that is in transit. This solves the problem of exposure.

Because intermediate routers do the retransmissions, recovery latency is reduced. However, this approach incurs additional burden on the routers, which are traditionally considered as store and forward nodes.

In our approach, we use an effective communication between the participating receivers. We explicitly provide all the receivers the necessary information about the topology (path to the sender) of the multicast session using a new *Sender-Initiated Multicast Trace-Route* mechanism as well as information about nearby participating receivers using a *Limited Scope Receiver Information Delivery* mechanism. This reduces the overhead put on the routers by PGM [8], LMS [7], ARM [9] and routers are freed from the additional burden of taking part in the loss recovery process of the end hosts. Since the receivers receive information about the nearby receivers as well as the path to the sender they group themselves to recover from the loss with a minimum recovery latency. Also the routers do not have to do caching of data as done by ARM [9] and do not have to generate the packets like NACK confirmations (NCFs) as done in PGM [8], do not have to keep the loss statistics of the attached links as done in LMS [7]. In our approach, the retransmission of the lost packets is done from a nearby receiver which has the requested packet which is determined by the use of WAIT packets and routers do not take part in the loss recovery process.

2.4.4 Resilient Multicast Protocol

Different multimedia applications like Video-On-Demand and Video-conferencing over the Internet may require IP Multicast service. Since real-time delivery of packets is essential for continuous playback of audio/video streams, it has been believed that attempting to recover from lost packets is not important or even not feasible. Instead, it has been observed that multimedia applications can tolerate certain amount of packet loss, hence most of the research is concentrated on devising adaptation techniques to minimize effect of packet loss and variable delays. But since multimedia applications already consume a lot of bandwidth such techniques add to the bandwidth utilization. Some *resilient* protocols have been developed that allow retransmissions to achieve higher quality. STORM[10] is

one such protocol.

Structure Oriented Resilient Multicast (STORM)

STORM [10] is a resilient multicast protocol in which group participants self-organize themselves into a dynamically built distribution structure and use the structure to recover lost packets from adjacent nodes. STORM is developed with two features of multimedia applications in mind:

1. minimize overhead of control packets since multimedia applications already consume large amount of bandwidth.

2. minimize delay in recovery of packets since packets arriving late will be discarded.

STORM distributes NACKs and repair packets along the structure. Each receiver selects its parent node depending upon the quality of the parent node in terms of loss experienced by the candidate parent node and the receiver's own playback buffer size. The playback buffer size determines how late can a packet as well as a repair arrive at the receiver and still be useful. When a receiver detects a packet loss it selects a parent node from the parent list and sends a unicast NACK for that packet to the parent. If after a certain time out if the packet has not arrived, then it selects another parent and sends another NACK. This continues until the packet is recovered or there is no need to recover it any more since the recovered packet will be discarded. When a parent receives a NACK it immediately unicasts the repair packet to the child if the repair is available. Since the repair is unicast to the child if there are many nodes experiencing the same loss there will be more retransmissions, which reduce the effective bandwidth utilization and if downstream nodes choose same node as a parent it can flood the parent.

In STORM, since each receiver sends the request to its parent node, in case of global loss, some receivers which are experiencing the same loss will be receiving the request

packets from the other receivers. In our approach, a local replier receiver explicitly sends a special WAIT packet to the nearby receiver(s) attached to the same multicast capable router to which the replier itself is attached. This has the effect that the root of the loss subtree is found when a local replier near the root of the loss subtree does not receive any WAIT packet sent by its upstream replier. The decision to send the request is also decided upon the delay that the local replier is going to experience for the recovery of the packet from its upstream replier receiver. If the expected recovery latency for a repair is more than the play-out time for the packet, the replier simply does not send any request and the overhead of unnecessary repair packet injected in the loss subtree is avoided. We also make the receivers "smarter" by checking if the repair packet can be useful depending upon the playback buffer size of the receiver and the expected recovery latency for the lost packet.

In STORM, each receiver has to maintain information about all the participants in the session. This leads to more overhead on the session. In our approach, we are avoiding that each receiver need to store information about each other participant. We ask the receivers to store only the information of the receivers that are attached to the same multicast router as itself and other receivers attached to next upstream router. This reduces the communication between the receivers as the request for lost packet travels either at the same level in the tree or only one level up the tree.

As IP Multicast is a best effort service which uses UDP as the transport layer protocol, losses are bound to occur during a multicast session. To recover from losses, either sender based approach or receiver based approach can be used. Sender based approaches have the problem of recovery latency as well as ACK Implosion. In receiver based approaches, receivers share the responsibility of helping other receivers to recover from the loss. However these approaches need to consider the *implosion* and *exposure* problems as discussed in Chapter 1. Most of the reliable multicast protocols recover each and every

packet. To avoid the implosion and exposure problems, most of the protocols either trade recovery latency or add more functionality in the routers. However, Continuous Media applications are loss tolerant but are delay sensitive. Hence, for such applications, the recovery of a lost packet should be done in real time to avoid unnecessary repair received after the packet becomes useless for the application due to real time nature of the data. Also, such applications do not need the fully reliable recovery provided by the protocols like SRM [6], PGM [8] and ARM [9]. STORM [10] is a resilient multicast protocol which takes into account the Continuous Media characteristics and tries to recover a lost packet till it concludes that the repair is going to be useless.

Chapter 3

Approach

Our approach of the protocol work is motivated by the argument that routers should be kept as simple as possible and they should not take part in the end-to-end issues like loss detection and recovery. In our approach, the end hosts perform loss recovery tasks of finding that a loss has occurred and recovering from the loss in the most effective way in terms of recovery latency and without the implosion and exposure problems (described in Chapter 1).

Our loss recovery technique works differently for the *Global Loss* and *Local Loss*:

- *Global Loss Recovery*

  If loss occurs on the 'trunk link' the receiver closest to the root of the loss sub-tree experiences the loss first. So it should be able to send the NACK to the receiver and/or sender whichever is closest to it and is above the loss link. The sender and/or the receiver should send the repair packet to the root of the loss subtree from where sub-casting of the repair to the loss subtree is done.

- *Local Loss Recovery*

  If loss occurs on the leaf link(s) then 'sub-casting' should not be done to avoid

unnecessary repair packets. Instead the repair should be unicasted. In this case, the loss should be able to be recovered by the co-ordination of the receivers closer to each other i.e. by some other receiver attached to the same multicast capable router or the next nearby receiver up the tree, avoiding other nearby receivers the overhead of processing unnecessary control packets.

Our approach, the WAIT protocol, tries to reduce the overhead on routers and tries to avoid the routers to participate in the end host issues. For reducing the overhead on the routers, the receivers can benefit from having some topology information that can be used during the loss recovery process. Receivers that know about their neighbors participating in the multicast session communicate with each other to recover from packet losses. To provide information to the receivers about its neighbors we use an efficient route tracing mechanism with minimum overhead on routers. Section 3.1 describes multicast traceroute mechanism in detail.

In order to have an efficient mechanism to recover from *global* and *local loss* (as discussed in Chapter 1), we observed that there should be an explicit communication between the receivers to decide the type of loss as well as the loss point in the multicast tree. For this, we elect some receivers as group heads depending upon their position in the tree and these group heads send a special WAIT packet to other designated receivers in its neighborhood. The WAIT packets function as the suppressors of the NACKs that can be sent by other nearby receivers to the group head which might lead to group head link being congested. To cater to the common loss experienced by some receivers below the loss point in an effective way, we use *sub-casting*. In *sub-casting*, the repair packet is unicasted to the root of the loss subtree which is the router that had dropped the packet previously, and this router then multicasts the repair packet to the other downstream receivers. Section 3.4 describes sub-casting in detail.

Our approach tries to develop a mechanism wherein not all the packets need to be

recovered, as this could still result in acceptable multimedia quality. Whenever the receivers record more loss than the decided session loss threshold, the receivers start the loss recovery process. Similarly, the decision to send the NACK and WAIT packets are decided based upon the loss percentage as well as the recovery latency expected and the playback buffer size of the receivers. Section 3.4 describes how the loss detection and recovery mechanism for our proposed WAIT protocol works.

3.1 Sender Initiated Multicast Traceroute

Typically, a receiver driven approach is used for tracing the route to the sender of any data packet. The receiver sends the traceroute packets towards the source in order to get the path to the source. However, in a multicast session, receiver driven traceroute will lead to overhead on the network as more and more packets will be injected in the network. The overhead increases linearly with the number of participants in a multicast session.

In our approach, the sender in the multicast session is responsible for the route tracing from itself to all the receivers. The sender sends a special traceroute packet in the multicast tree, which follows the same path along the tree setup by the multicast routing algorithm for the multicast session. When the multicast capable router along the path of the tree encounters such a packet it adds its address into the packet. The packet is then forwarded through all the interfaces leading to the other receivers participating in the multicast session and/or other multicast capable routers that are along the multicast path to the participants. Figure 3.1 shows this concept. Each receiver participating in the session need not trace path to the source individually, saving overhead on the network in the case of many receivers.

By using the information in the traceroute packet all the receivers participating in the multicast session have the path information which can be used for our proposed loss

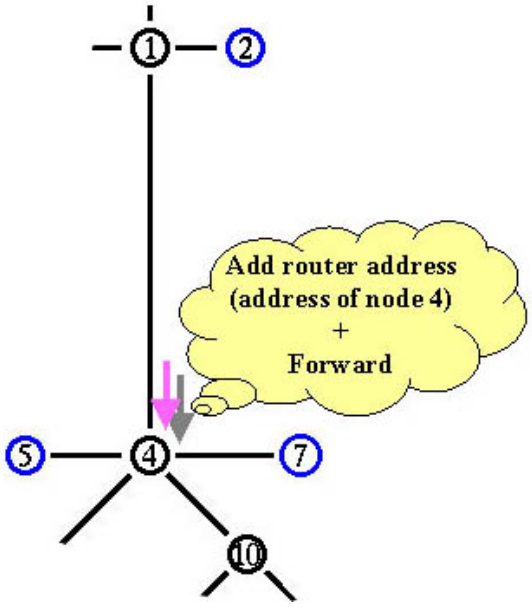

Figure 3.1: Multicast Traceroute Mechanism

recovery approach. The information that all receivers need to know for the loss recovery made available by the traceroute packets is:

1. All the multicast routers present along the path to the source from the receiver.
2. The multicast router to which the receiver is attached directly or indirectly using the unicast routers.

Receiver	Path From Source	Nearest Router	Upstream Router
2, 3	1	1	-
5, 7	1, 4	4	1
8, 9	1, 4, 6	6	4
11, 12	1, 4, 10	10	4

Table 3.1: Receiver Topology Information

For the multicast session topology shown in Figure 1.1, the information about the topology that is made available to the receivers participating in the session, is summarized

in the Table 3.1. The same traceroute packet is also used to find the round-trip-time (RTT) between sender and receiver by including a time-stamp in the packet.

3.2 Limited Scope Receiver Information Sub-Cast

The receiver receiving a traceroute packet finds out to which multicast capable router it is directly attached. The receiver then sends a ping request to the multicast router to which it is attached to find the RTT to that router from the receiver. Once the receiver gets the information about the RTT to the router it then sends an information packet to the router to sub-cast the information using tunneling mechanism to the downstream receivers attached to the next downstream multicast capable router to which at least one receiver is directly attached.

This information packet has the source address as the address of the sender of the multicast session and the destination address as the group address of the multicast session.

In short the information packet has the following information:

1. Source address = Session Sender Address.
2. Destination address = Session Group Address.
3. RTT to the multicast capable router to which the receiver is attached.
4. The time the information packet is sent.
5. The quality this receiver is maintaining for the session.

The scope of these information packets are limited using time to live (TTL) field in the packet header. This reduces the problem of flooding the packet to all the downstream receivers as the farthest receiver will have to process many information packets. By avoiding flooding we avoid extra overhead on the receivers. The only receivers that

require information about the sender of the information packet are the receivers which are immediately downstream to it as well as receivers attached to the same multicast capable router to which the sender of the information packet is attached for our designed protocol to work.

When a receiver receives an information packet from the other nearby receivers attached to the same multicast capable router as well as other upstream participating receiver(s), it stores the information for later selecting the best replier for the loss recovery of packets. The information that a receiver gets from an information packet can be summarized as:

1. The address of the sender of the information packet
2. The RTT of the sender to its multicast capable router to which it is directly attached in the multicast tree.
3. RTT between the sender of the information packet and the receiver receiving the information packet.
4. The quality that the sender of the information packet is going to maintain if it becomes the replier for other receivers.

After the receivers receive all the information packets they decide which is the best replier receiver for them as described next in Section 3.3.

3.3 Replier Receiver Selection

Finding the best replier for the recovery of lost packets in a multicast session has the following goals:

1. Low recovery latency to the replier.

2. Directing the request only to the receiver that has successfully received the requested packet.

3. In case of global Loss, finding the root of the loss sub-tree for the starting point from where the repair packets will be sub-casted.

4. In case of global Loss, *sub-casting* the repair packets only from the root of the loss sub-tree so that only the receivers in the loss sub-tree receive the repair packet.

5. Avoiding the unnecessary NACKS in the loss sub-tree, as these NACKs result in duplicate repairs being sent in the loss subtree.

6. Allow only one NACK to escape from the loss sub-tree, to avoid an explosion of repair packets in the loss sub-tree leading to duplicate repairs.

After the receiver(s) receive the information packet(s), the receiver(s) check which is the closest receiver to the multicast capable router to which it itself is attached. The receiver closest to the multicast capable router (in terms of RTT to the router) acts as the group head for the other receivers attached to the same multicast capable router and is responsible for the recovery of lost packets experienced by the other receivers attached to same multicast router. This replier acts as the *local replier* for the other receivers attached to the same multicast capable router. And the same receiver asks for the retransmission of any packets lost on trunk links (i.e. *global loss*) in the tree as described later.

The local replier also checks for the upstream closest receiver by using the information about the next upstream multicast router and the closest receiver attached to that router. The decision about which is the closest receiver is done using the router-RTT received in the information packet sent by the upstream receiver(s). This upstream receiver acts as the replier for the downstream local replier receiver. The local replier receiver sends a message to the upstream replier about its existence in the session. The upstream receiver

that is actually the local replier receiver for the upstream group of receivers keeps the information about the next downstream local replier receiver.

3.4 Loss Detection And Loss Recovery Process

Each receiver keeps track of the sequence number of the packets that the sender sends to the multicast session. When the receiver receives a packet sequence number greater than the next expected packet sequence number it concludes that the packet is lost and starts the loss recovery process.

The loss recovery process in the WAIT protocol is divided into two parts depending upon the approximate point of loss of the packet in the multicast tree. The point of loss is difficult to determine and at the same time important in order to avoid unnecessary NACKs and repairs. The loss can occur on either the trunk link(s) (*global loss*) or the leaf link(s) (*local loss*) as described in Chapter 1.

3.4.1 Case1 : *Global Loss Recovery*

In this case, the loss occurs on the trunk link. For example, for the topology shown in Figure 1.1, if the loss occurs on the link 1-4, all the receivers (5, 7, 8, 9, 11, 12) down the link experience that loss. The receiver closest to the loss, receiver 5, experiences the loss first as it is the closest receiver to router 4. Each receiver starts its recovery mechanism to recover from the loss. As described earlier, receivers know about their own replier receivers as well as the responsibility that the receivers might have to carry for other local receivers. Table 3.2 gives an example of the relationship between the receivers. The table can be read as :

1. For receiver 5, 7 is the replier receiver and

Receiver	Replier Receiver
2	0
3	2
5	2
7	5
9	5
8	9
11	5
12	11

Table 3.2: Receiver-Replier Relationship

2. For receiver 12 , 11 is the replier receiver and so on.

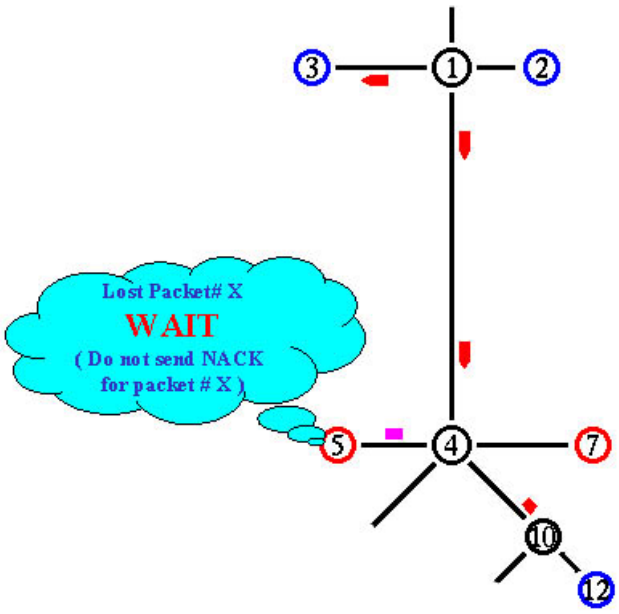

Figure 3.2: Loss Detection and WAIT Mechanism

As receiver 5 experiences loss occurred on link 1 - 4 first, it sends a special WAIT packet to all the local receivers i.e. receivers attached to the same router as well as the next downstream local replier receiver. Similarly, receiver 9 sends a WAIT packet to

Receiver Sending WAIT	Receiver(s) receiving WAIT
2	3, 5
5	7, 9, 11
9	8
11	12

Table 3.3: WAIT Packet Flow

receiver 8 and so on, as shown in Figure 3.4.4. Table 3.4.4 summarizes the flow of WAIT packets to the receivers.

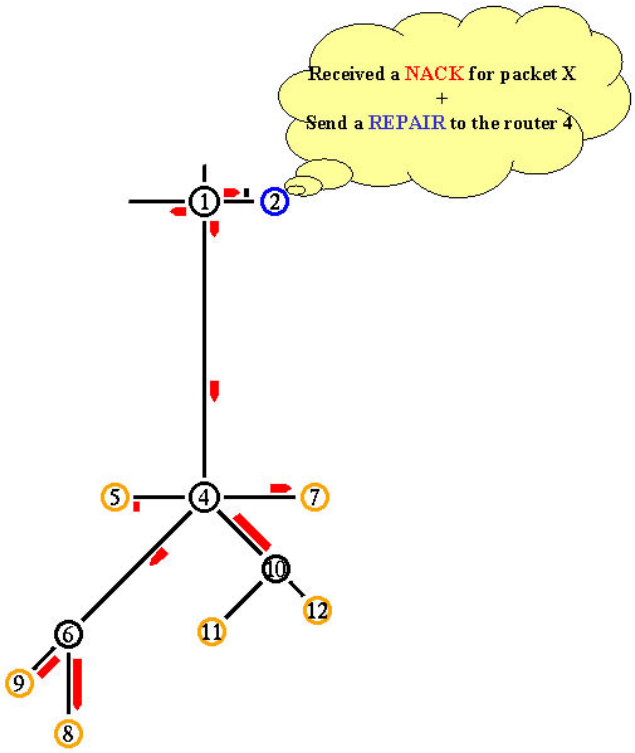

Figure 3.3: Nack Receipt at the Replier

When the receivers receive the WAIT packet sent for the sequence number(s) it does not send any NACK to its replier receiver. The loss in this case has occurred on link 1-4 and hence receiver 2 has received the packet and it does not send any WAIT packet to receiver 5. As the receiver 5 does not receive any WAIT packet from its replier receiver 2,

it determines that it is the closest receiver to the point of loss. Each local replier receiver (e.g. 2, 5, 9, 11) waits for the amount of time equal to the RTT between the replier receiver and itself. When a local replier receiver does not receive any WAIT packet for the lost packet(s), it decides to send the NACK to the upstream replier receiver. In this case, receiver 5 does not receive any WAIT for the lost packet(s) so it sends a NACK to the upstream replier receiver (receiver 2) as shown in Figure 3.3.

The NACK sent by a *local replier receiver* consists of the following information:

1. Sequence number of the lost packet that is to be retransmitted.
2. The address of the multicast capable router which is the root of the loss sub-tree.
3. In case of bursty loss, the starting sequence number of the loss burst and the burst length.

When a replier receiver receives a NACK sent by the next downstream replier receiver, it checks for the availability of the packet with it. If it finds that the requested packet is available with it, then it *encapsulates* the requested packet into another packet addressed to the router which is assumed to be the root of loss subtree. This router address is informed by the sender of the NACK to the replier receiver. In our example, receiver 5 informs receiver 2 to send the repair to the router 4.

The repair packet is then sent to the designated router. The router with its address as the destination address of the packet then strips off the outer packet and then sub-casts it down the tree as shown in Figure 3.4.

In our example, when loss occurs on the link 1-4, the following sequence of events take place:

1. Receiver 5 waits to receive any WAIT packet from the upstream receiver 2. Since receiver 2 has not experienced any loss it does not send any WAIT packet to receiver 5.

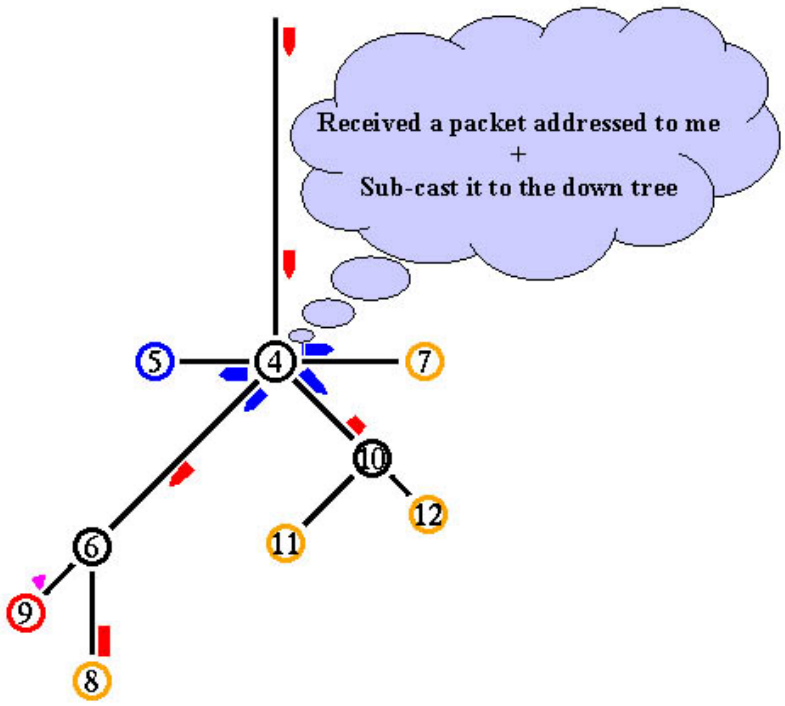

Figure 3.4: Router Sub-cast Mechanism

2. Receiver 5 does not get any WAIT packet from receiver 2 after the expiration of the time of one RTT to the receiver 2. Receiver 5 concludes that loss has occurred on link 1-4 and sends a NACK to receiver 2 with the lost sequence number and the address of the router 4.

3. Receiver 2 receives the NACK and checks for the availability of the packet.

4. If it finds the requested packet in its play-back buffer, it encapsulates the repair packet in a packet with destination address of router 4. The inner packet has its source address as the source address of the session (address of node 0) and the destination address as the group address of the multicast session.

5. Receiver 2 sends the encapsulated repair packet to the router 4.

6. Router 4 receives the encapsulated packet and decapsulates it.

7. Router 4 looks at the source and destination address of the inner packet and forwards the packet on the appropriate interface leading to other receivers in the loss subtree.

8. The downstream routers 6 and 10 deliver the repair packet as if it is a normal session packet.

3.4.2 Case2 : *Local Loss recovery*

In case of loss occurring on the links other than the 'trunk links' in the tree, the working of WAIT protocol is different.

The receiver (receiver(s) other than the replier receiver(s)) experiencing the loss first waits for its local replier receiver to send a WAIT packet. When the receiver finds that the local replier has not sent any WAIT packet for the same loss after waiting for one RTT to the local replier, it concludes that the loss has occurred on its link to the router and also concludes that the lost packet is available with the local replier receiver and sends a NACK to the local replier asking for the lost packet.

A local receiver receiving a NACK sent by its neighboring receiver checks for the availability of the requested packet with it. If it is available the local replier then unicasts the repair packet to the requester. Here no sub-casting mechanism is used and hence the NACK does not contain any router address.

The following events occur in case of *Local Loss Recovery* when loss occurs on link 4-7:

1. Receiver 7 detects the loss and waits for the WAIT packet being sent by its local replier receiver 5.

2. Receiver 7 does not receive any WAIT packet from the receiver 5 and concludes that the loss has occurred on link 4-7.

3. Receiver 7 sends NACK to its local replier receiver 5.

4. Receiver 5 receives the NACK sent by the receiver 7 and checks for the availability of the packet with it.

5. Receiver 5 unicasts the repair packet to the receiver 7.

6. Receiver 7 receives the repair packet and finishes the loss recovery process for the lost packet.

3.4.3 Special Case: *Replier Link Loss*

If loss occurs on a replier link then the replier receiver incorrectly assumes that the loss has occurred on the trunk link and starts the *Global Loss Recovery* process. The receiver sends a WAIT packet to the nearby receivers and also waits to receive any WAIT packet from the upstream receiver. When it does not receive any WAIT packet from its replier receiver it incorrectly concludes that the loss has occurred on the trunk link and sends a NACK to its upstream replier receiver. Here two cases are possible:

1. ***Neighbor Receiver Closer than the Upstream Replier Receiver***:

   In this case, a receiver is closer to the replier receiver than the upstream replier receiver in terms of RTT as shown below:

   RTT / 2 of the neighbor $<$ *RTT / 2 of the upstream replier receiver.*

   The neighbor decides to send the repair as it has determined that its reply is going to reach the local replier receiver before it times out and sends a NACK to the upstream replier.

In our example, receiver 5 experiences a loss (because of loss on link 4-5) and sends a WAIT to other nearby receivers. Receiver 7 receives the WAIT packet. It looks into its table of neighbors and finds out that it is closer to the local replier receiver (receiver 5) than the upstream replier (receiver 2) and hence decides that a repair packet sent by it will be delivered to the receiver 5 before receiver 5 times out and sends a NACK to receiver 2. Therefore, receiver 5 sends a repair packet to receiver 2. Receiver 2 receives the repair and does not send a NACK to receiver 2 asking for sub-casting the repair packet to the incorrectly assumed loss tree by the receiver 5. This avoids the *exposure* problem, in the case of replier link loss.

2. ***Neighbor Receiver farther than the Upstream Replier Receiver***

   In this case, a receiver is farther to the replier receiver than the upstream replier receiver in terms of RTT as shown below:

   RTT / 2 of the neighbor $>$ *RTT / 2 of the upstream replier receiver.*

   In our example, if receiver 5 (the local replier) experiences loss (because of loss on link 4-5) and sends the WAIT packet, receiver 7 receives the WAIT packet and checks if the packet is available with it. It also checks the RTT between the local replier and the upstream replier and also the RTT between itself and the local replier. If it finds that the upstream replier is closer to it than itself to the local replier then it decides not to send any repair as receiver 5 is going to timeout before the repair packet sent by it reaches receiver 5. To improve the recovery latency of receiver 5, receiver 7 might be asked to send the repair packet as it will reach before the repair sent by the upstream replier, thus all the receivers down the router 4 are going to get duplicates if this approach is taken. In this case we can not avoid the *exposure* problem.

3.4.4 Resilient WAIT

Above we discussed about recovering each and every lost packet by using our approach. However, we pointed out in Chapter 2 that multimedia applications can tolerate some amount of packet loss but are delay sensitive. We make use of these characteristics of multimedia applications to configure our protocol to work effectively for multimedia multicast applications.

Before sending the WAIT or NACK packet each receiver checks the amount of loss that it has experienced. If the amount of loss is below the tolerable amount of loss by the multimedia application, then the receivers do not send any WAIT or NACK packets. Also, if the replier receivers find that the expected recovery latency for the lost packet is more than its playback buffer size, they do not send any WAIT or NACK packets.

WAIT is designed to be a tunable resilient multicast protocol which can be used for different quality requirements of multimedia multicast sessions. Our approach gives more emphasis on local recovery by requesting the retransmission from a nearby receiver. The receiver select replier receivers according to the information they get from the topology (Section 3.3). By using the WAIT packets finding the root of loss sub-tree without the network support is possible. And in case of global loss the replier receiver sends an encapsulated packet to the root of loss sub-tree from where the repair is forwarded using normal IP Multicast forwarding mechanism (Section 3.4.1). In case of local loss, WAIT recovers the lost packet using unicast mechanism from a local replier receiver attached to the same multicast router to which the receiver experiencing the loss is attached (Section 3.4.2). This reduces the unnecessary requests and repairs seen by other receivers. WAIT adds more functionality in the receivers than in the routers which is the case for some of the

router assisted protocols like PGM [8] and ARM [9]. Using the Continuous Media characteristics (loss tolerance and delay sensitivity) WAIT reduces overhead on the session by not requesting any lost packet till the receivers experience loss above the tolerable loss threshold for the session or if the receivers conclude that the repair packet is going to be useless because of real time nature of the data (Section 3.4.4).

Chapter 4

Simulation Design and Implementation

For studying our proposed approach, WAIT, for loss recovery in multimedia multicast sessions, we used an event driven network simulator NS2 (Network Simulator - Version 2) [11]. NS2 has gained acceptance in the research community because of its rich support for the IP network components like TCP (Tahoe, Reno, Vegas) and UDP agents. NS2 also supports multicast. SRM [6] is also incorporated into NS2. NS2 has a rich API for developing various types of agents that can be attached to the IP network nodes to develop new protocols and also supports a Constant Bit Rate (CBR) traffic generator which we used in our performance studies.

4.1 Implementation

We have designed and implemented three agents for the WAIT protocol: *Wait-Sender*, *Wait-Agent* and *Wait-Receiver*. All these agents developed are derived from the base class *Agent* in NS2. We also developed a new *WaitErrorModel* for controlled simulation of loss.

4.1.1 Wait-Sender Agent

A Wait-Sender agent sends application DATA packets as well as the trace-route packets in the multicast tree. Before sending the DATA packets this agent adds its own protocol specific headers to the packets. Each time a DATA packet is sent at some constant interval of time, it adds to its header a sequence number for the data packet which is used by the Wait-Receiver agent to detect loss using gap-detection mechanism.

4.1.2 Wait-Receiver Agent

The core functionality of the working of the WAIT protocol is designed and implemented in this agent.

To add the functionality of a multimedia client in our design, we implemented a *play-back buffer* mechanism in this agent. The playback buffer is used to keep track of the packets received, packets lost, packets repaired as well as the maximum delay that the receiver should tolerate for the recovery of the lost packet. This playback buffer is also used to collect the statistics after the simulation run. The Wait-Receiver agent performs the following functions depending upon the type of packet it has received:

1. WAIT-TRACE-ROUTE packet:

   When a Wait-Receiver agent receives a TRACE-ROUTE packet sent by the sender, the agent stores the path. It then finds the last address of the multicast router which is the address of the multicast router to which it is attached.

2. WAIT-PING packet:

   When a Wait-Receiver receives a PING packet, it calculates the RTT to the nearest multicast router and stores it. Then the Wait-Receiver sends an information packet containing information about itself, as described in the Section 3.2, to other down-stream receivers.

3. WAIT-RECEIVER-INFO packet:

   When the Wait-Receiver agent receives a RECEIVER-INFO packet, it stores the information about the receiver in a table which it uses later on to decide the best replier for the receiver as described in the Section 3.3.

4. WAIT-DATA packet:

   When a Wait-Receiver receives a DATA packet, it simply hands the DATA packet to the playback buffer to check for any lost packet. If the playback buffer informs the receiver that a loss has occurred, it checks if the loss experienced by the receiver is above the quality limits set for the session. If the loss is below the threshold, it does not start the recovery process. If loss is above the threshold, it first starts the process of replier selection (as described in Section 3.3) and then it starts the recovery process (as described in the Section 3.4). It starts a NACK timer to send the NACK for the loss of packet(s) depending upon the delay between the receiver and its replier receiver.

5. WAIT-WAIT packet:

   When a Wait-Receiver receives a WAIT packet, it checks if there is any outstanding NACK timer being set to fire a NACK for the sequence number(s) sent using the WAIT packet. If there is any NACK timer pending for the sequence number(s), the receiver cancels that NACK timer. If the receiver does not receive any WAIT packet before the RTT between itself and its replier, it sends the NACK asking for the repair of the packet using either unicast or sub-cast mechanism as described in Section 3.4

6. WAIT-NACK packet:

   When a Wait-Receiver receives a NACK packet sent by the some other receiver, it

queries the playback buffer to check for the availability of the packet in the playback buffer. If the playback buffer informs about the availability of the packet it then either encapsulates the repair packet or sends it directly to the requester.

7. WAIT-REPAIR packet:

   When a Wait-Receiver receives a REPAIR packet, it hands that over to the playback buffer and also checks if the REPAIR packet was sent using either sub-cast or uni-cast mechanism. If a uni-cast mechanism has been used while sending the repair and if that receiver has received a NACK for the sequence number then it uni-casts/sub-casts (depending upon the type of repair requested) the repair to the requester.

4.1.3 Wait-Agent Agent

The Wait-Agent agent mimics the behavior of a multicast capable router attached to a node. It works on top of the other multicast routing protocols being implemented in NS2. The multicast routing protocols use the Nodes API for their functionality. If a packet is not of a type of packet sent by the Wait-Sender, then the Nodes API forwards the packet using the routing tables used in the nodes. When a Wait-Agent agent receives a packet of type which is WAIT specific it carries out following functionalities:

1. WAIT-DATA packet:

   When the Wait-Agent receives a DATA packet it checks if the error flag in the packet is set to 1 by the previous Wait-Agent that has seen this DATA packet before this agent. If the error flag is set to 1, it concludes that the packet should not be forwarded down the tree and it drops the packet creating loss on the link on which it receives the packet. If the error flag is not set to 1, the Wait-Agent hands over the

packet to the node to which it is attached so that the node can forward the packet on the interfaces leading to the other participants in the multicast tree.

2. WAIT-TRACE-ROUTE packet:

   When the Wait-Agent receives a TRACE-ROUTE packet, it simply adds its address (the node address) in the packet and forwards that packet on all links leading to other participants in the multicast session as shown in the Figure 3.1.

3. WAIT-PING packet:

   When the Wait-Agent receives a PING packet sent by the receiver, it simply sends that same packet back to the sender of the PING packet.

4. WAIT-RECEIVER-INFO packet:

   When a Wait-Agent receives a RECEIVER INFO packet it checks for the time to live (TTL) value in the packet header. If the value is found to be 0 it drops the packet. If the value is not 0, it forwards the packet on the links leading to other participants down the tree.

5. WAIT-WAIT packet:

   When the Wait-Agent receives a WAIT packet, it forwards the packet to all the receivers attached to the router to which it is attached.

6. WAIT-NACK packet:

   When a Wait-Agent receives a NACK packet, it forwards that packet to the underlying forwarding mechanism which sends the NACK packet to the destination address of the packet.

7. WAIT-REPAIR packet:

When a Wait-Agent receives a REPAIR packet, it checks whether the destination address of the packet is same as its own node address. If the addresses are the same, it concludes that it has received an encapsulated packet and hence decapsulates the inner packet. It then forwards this inner repair packet to the links leading to the other downstream participants in the multicast session using the forwarding mechanism of the node to which it is attached as shown in Figure 3.4.

4.1.4 WaitErrorModel

In NS2, losses can be simulated by using different implementations of the error-models (list error-model, Select error-model, and Periodic error-model to name a few). But none of the error-model provides the necessary functionality to simulate losses of some predefined loss percentage.

For studying the effects of bursty as well as random losses on the links depending upon the quality set by the receivers, we implemented a new WaitErrorModel, which drops packets on the links depending upon the loss percentage set by the simulation setup. Whenever, the loss on the link is below the desired loss, the error-model sets the error flag in the header of the DATA packets to 1, indicating that this packet needs to be dropped for simulating the loss.

4.1.5 Miscellaneous

In the case of two or more receivers having exactly the same delay (which is usually not the case in the real world) to a same attached router in the simulation and if these receivers have a race condition for the local replier position, we use the address of the nodes to break the tie. For example, if receiver 5 and receiver 7 have the same delay to router 4 then the receiver which has the lower node address acts as the replier receiver.

4.2 Simulation Setup

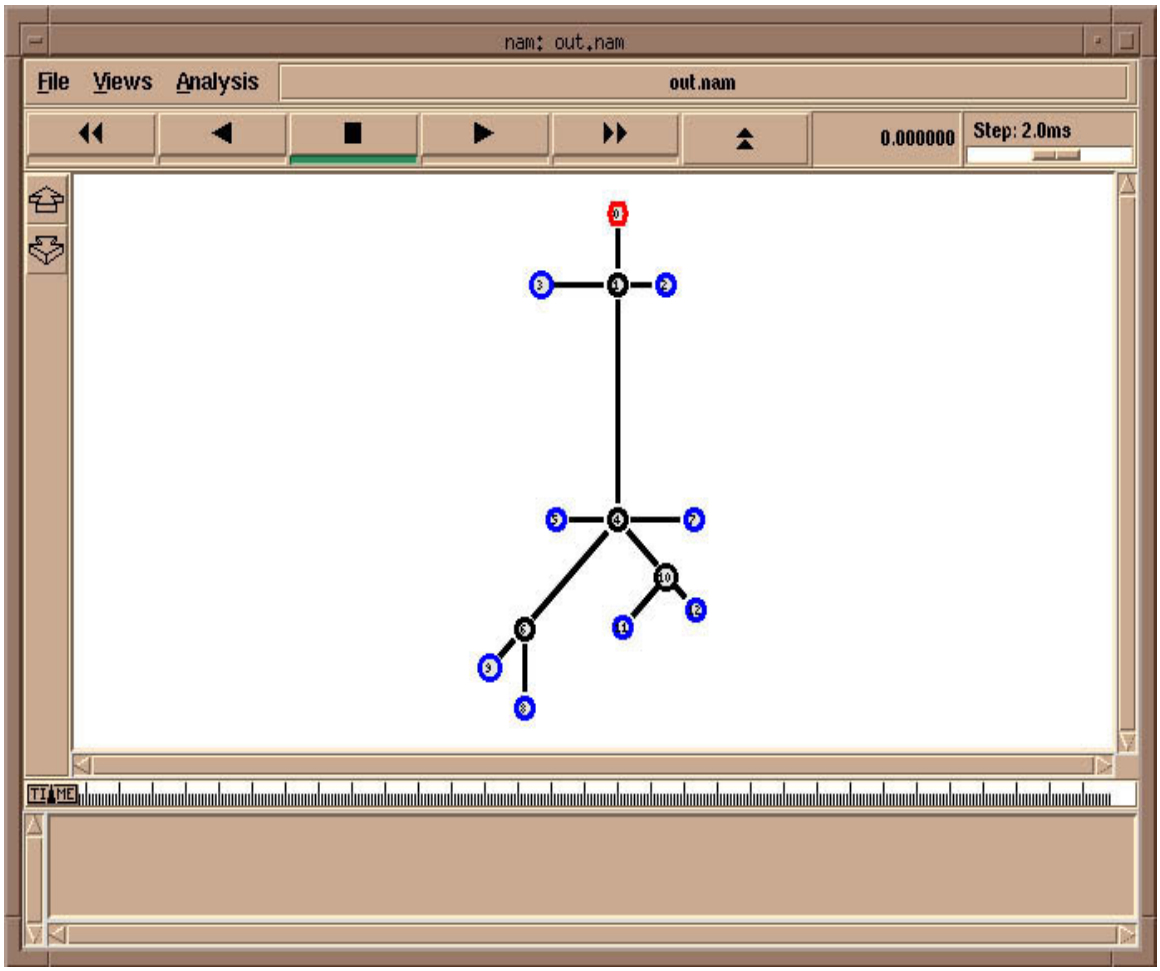

Figure 4.1: Snapshot of Network Animator Tool (NAM) and the Topology Used for the Simulation Runs.

To carry out the performance study of our approach, we carried out a number of simulation runs using the WAIT protocol agents, SRM [6] agents as well as PGM [8] protocol agents. In order to study the effects of loss and the loss recovery mechanism we used four different configurations of WAIT protocol agents. We configured the agents to maintain the desired quality of 100%, 98%, 95%, and 90% for the simulation runs. Here, quality for 100% configuration means that all lost packets need to be recovered

	Minimum Delay (msec)	Maximum Delay (msec)
1X	2	25
2X	4	50
5X	10	125

Table 4.1: Minimum and Maximum Link Delays (msec) used during the simulation runs.

and quality for 95% means the agent allows for 5% packets to be lost and do not ask for the repair for 5% lost packets. We used a 100% configuration to compare WAIT with fully-reliable protocols PGM [8] and SRM [6]. Other configurations were used to study the effects of WAIT by considering Continuous Media characteristics on the network. We used a maximum of 10% tolerable loss since usually tolerable loss for an audio stream is around 5%. We created a topology as shown in Figure 1.1. For the evaluation and debugging purpose we used the Network Animator (NAM) tool as shown in Figure 4.1. The tool proved to be very useful during debugging the protocol actions.

The Wait-Sender agent is attached to node 0, Wait-Agent agent is attached to nodes 1, 4, 6 and 10. The Wait-Receiver agents are attached to nodes 2, 3, 5, 7, 8, 9, 11 and 12. The Wait-Sender agent was configured to mimic the behavior of an audio stream. The Wait-Sender Agent sends a DATA packet every 160 milliseconds (audio packets of 20 milliseconds sampled at 8,000 Hz).

To study the effect of different link delays on the average recovery latency as well as end-to-end latency experienced by the agents, we added maximum and minimum delays for the links in the topology as shown in Table 4.1.

To study the effect of various link loss percentage on the overhead experienced by the agents, we added 2%, 4%, 6%, 8% and 10% link loss percentages on each link. Table 4.2 summarizes per-link loss percentage and the maximum loss experienced by any receiver in the simulation run.

In order to study the behavior of different WAIT agent flavors we ran the simulation

Loss Per Link (Percent)	Minimum Loss (Percent)	Maximum Loss (Percent)
2	4	6
4	8	12
6	12	18
8	16	24
10	20	30

Table 4.2: Minimum and Maximum Link Loss (Percent) experienced by any receiver during the simulation runs.

tests on all the combinations of the above described scenarios using different per link loss percentage and link delays. We also ran the same tests for SRM [6] and PGM [8] agents.

4.2.1 Simulation Run Time Selection

In order to get stable statistics for the above mentioned combinations of the agents as well as network scenarios, we ran the simulations for different durations as shown in Figure 4.2 and studied the average recovery latency (Seconds) reported by the protocol agents. We found that for 10 and 50 second duration of the simulation run, the results were not uniform, as showed by the variation in the recorded values. For the simulation runs of 100 and 500 seconds we observed that the values remain almost constant as indicted by the values being almost same for different configuration of the WAIT protocol agents. Therefore, we concluded that the duration for the simulation run can safely be kept at 100 seconds in order to get stable statistics.

We ran the simulation tests for 100 seconds duration for all the above mentioned combinations of topology and protocol agents. The next chapter (Chapter 5) describes the results and analysis of the simulation runs.

We implemented our approach in an event driven simulator NS2. We developed three agents the Wait-Sender, Wait-Receiver and Wait-Agent agents to simulate the functionality of the WAIT protocol (Section 4.1). We also developed an error-model which sim-

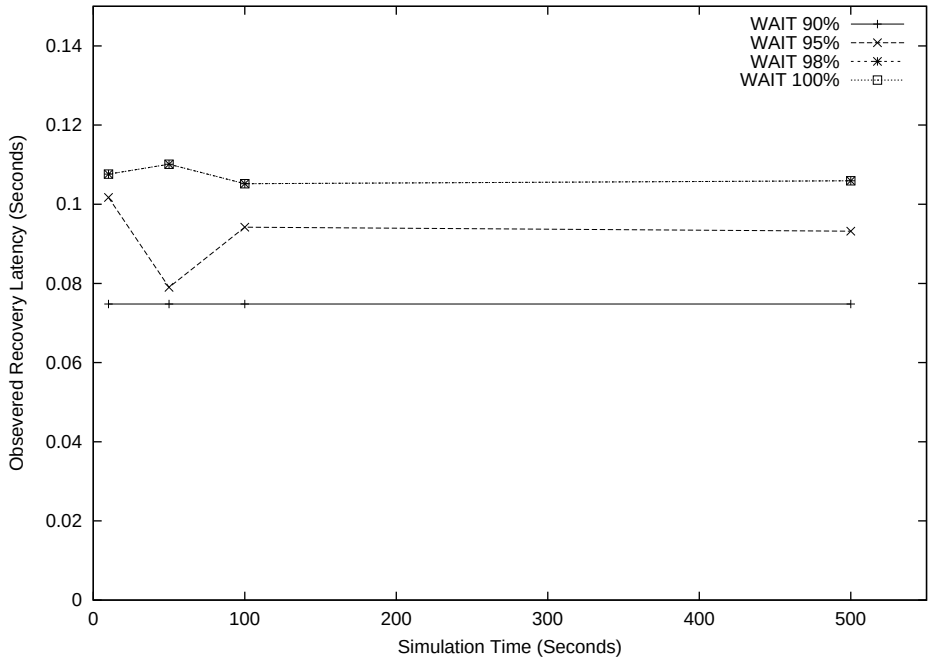

Figure 4.2: Simulation Run Time Determination: Graph of Recovery Latency (Seconds) *versus* Simulation Duration (Seconds)

ulates loss of desired percentage on the links in the topology (Section 4.1.4). Different topologies and scenarios were simulated using different link delays and per-link loss percentages to study the effects of loss and delays on the WAIT protocol (Section 4.2). We ran our simulations for different durations to decide for the duration of the simulation runs (Section 4.2.1).

Chapter 5

Simulation Results and Analysis

We present a comparison of our proposed WAIT protocol using the metrics mentioned below with SRM [6] and PGM [8]. We also describe the overhead that the receivers experience because of our approach. We also present results that can be used by multimedia application with different quality requirements in terms of loss allowed as well as recovery and end-to-end latency.

For evaluation purpose we use following metrics:

- Average Recovery Latency: The average of the delay that each receiver experienced from the loss detection till the arrival of the repair packet.
- Average End-to-End Latency: The average end-to-end latency experienced by the session participants due to the recovery process.
- Percentage Overhead: The overhead in terms of the control messages being sent for the recovery process.
- Percentage Recovery: The quality maintained by the receivers for different configurations of the WAIT protocol. It determines the percentage of packets that has been recovered during the session.

5.1 WAIT Performance

First, we present the performance of the WAIT protocol for a 98% quality requirement for a multicast session (this means that the agents can tolerate 2% of packet loss). We achieved this by setting the desired quality that should be maintained by the *Wait-Receiver* agents to be 98%.

5.1.1 WAIT Performance - Fixed Link Delays

For studying the behavior of our proposed WAIT protocol with a 98% quality requirement on a particular topology, we simulated loss of 2%, 4%, 6%, 8% and 10% on each link using the topology 2X with minimum and maximum per-link delays set as shown in Table 4.1

WAIT Overhead

We calculated the overhead experienced by the receivers because of the WAIT protocol and then took the average of it to decide the overall session overhead. As shown by the graph in Figure 5.1, the overhead grows linearly as per-link loss is increased. With the overhead being near to 7.5% for 2% loss on each link and about 38% for 10% loss on each link.

WAIT Average Recovery Latency

In the graph in Figure 5.2, we present the average recovery latency experienced by the receivers in the session. For the per-link loss of 2% the average recovery latency is around 0.1 seconds. Whereas, as the per-link loss is increased, the recovery latency also increases to around 0.13 - 0.14 second. We found that this is mainly because of the increase in the per-link loss on the trunk link 1-4, which is the high latency link in the currently studied

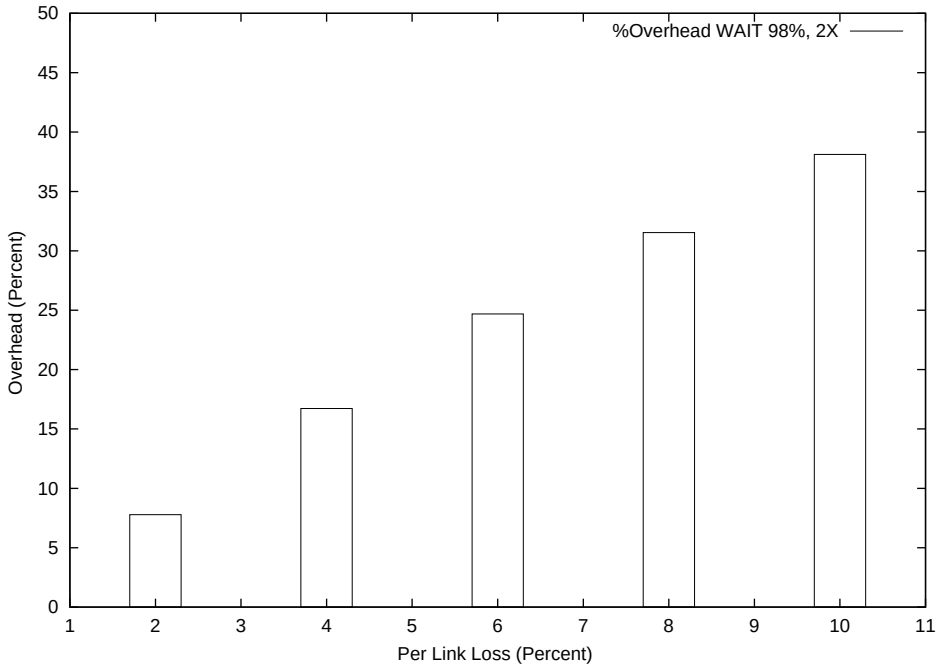

Figure 5.1: WAIT 98% : Overhead *versus* Per-Link Loss Percentage for *2X* Topology

topology of 2X. Since more and more packets on this link are recovered, the average recovery latency keeps on increasing.

WAIT Recovery Percentage

We also studied the actual percentage of the lost packets being recovered. As shown by the graph in Figure 5.3, we found that the actual percentage of packets being recovered for the entire session is *almost* same as that of the requested quality of 98%. Observing the graph it is clear that as the per-link loss is increased to 10%, the actual recovery percentage is reduced to around 96%. We found that this is due to a cumulative effect of packets getting lost and the quality maintained by the replier receivers. We need to study this cumulative effect in order to understand its implications on the quality maintained. We intend to study this effect as future work.

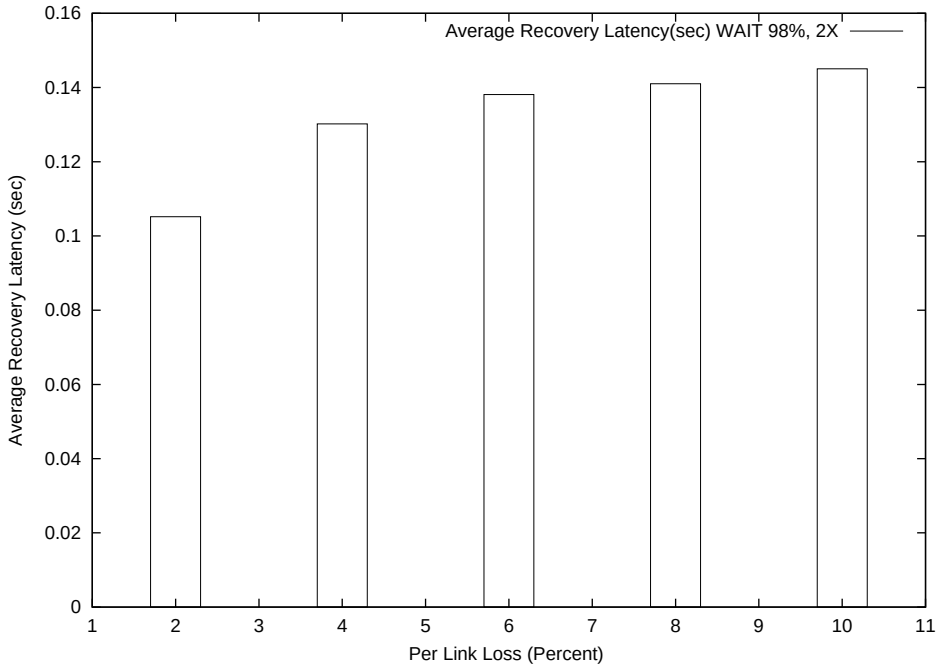

Figure 5.2: WAIT 98% : Recovery Latency *versus* Per-Link Loss Percentage for *2X* Topology

5.1.2 WAIT Performance - Fixed Per Link Loss Percentage

For studying the behavior of our proposed WAIT protocol with 98% quality requirement with same per-link loss percentage, we simulated loss of 4% on each link for the three topologies using the link delays set to 1X, 2X, 5X with minimum per-link delays set as shown in Table 4.1

WAIT Overhead

The graph in Figure 5.4 shows the overhead incurred by the WAIT protocol for a 98% quality requirement for the three topologies (1X, 2X, 5X) described in Table 4.1. The overhead is calculated using the ratio of all the packets that the receivers has received other than the DATA packets to the total number of packets sent by the sender. Observing the graph, we found that the overhead incurred for the session for 1X and 2X topologies

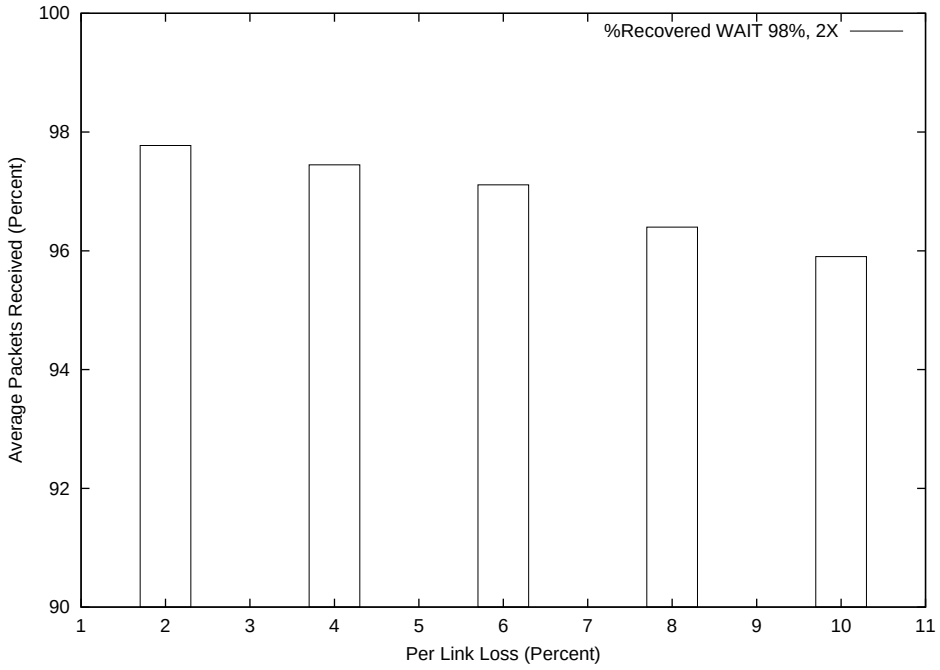

Figure 5.3: WAIT 98% : Recovery Percentage *versus* Per-Link Loss Percentage for *2X* Topology

is almost same and is around 16%. But as can be seen, the overhead for the 5X topology is reduced to around 8%. The main reason behind this reduction in overhead is that as the link delay is increased, the decision about whether to send the request to the replier depends upon the expected recovery latency. And as the link delays are increased this decision plays a prominent role in reducing the overhead as late arriving packets are going to be discarded due to real time nature of the multimedia applications.

WAIT Recovery Latency

The graph in Figure 5.5 shows the effect of the link delays on the average recovery latency observed by the session. The recovery latency is calculated by taking the difference between the time that a loss is detected and the time when the lost packet is recovered. As seen from the graph, the average recovery latency clearly depends upon the link delays

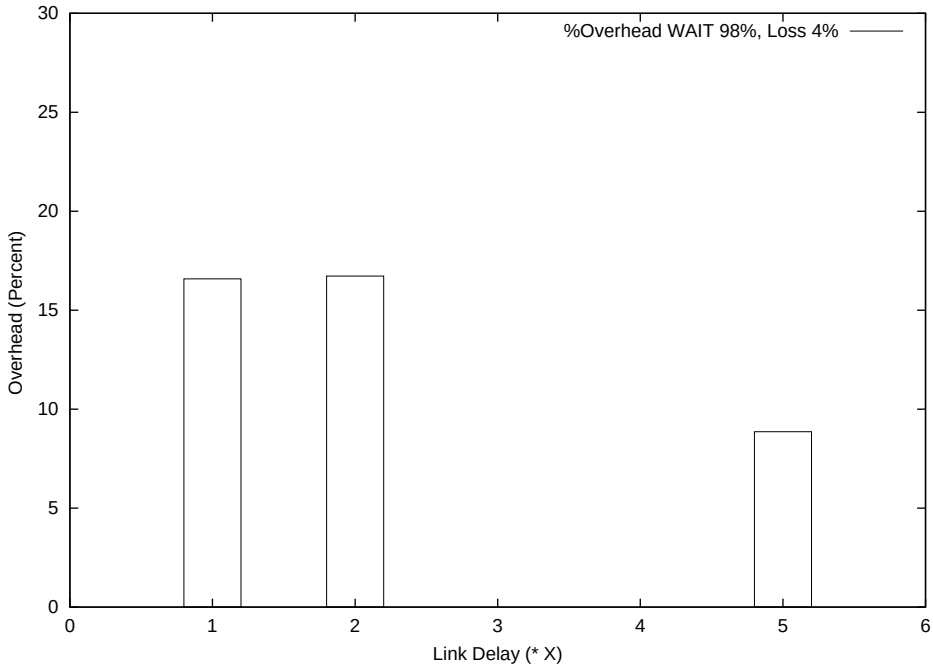

Figure 5.4: WAIT 98% : Overhead Percentage *versus* Link Delay for Per-Link Loss of 4%

and the decision to recover any lost packet which depends upon the playback-buffer size, which for the simulation runs we kept at 250ms. As seen, for a 1X topology, the average recovery latency observed by the receivers is around 0.07 seconds and for the 2X topology, the average recovery latency is around 0.13 seconds as expected as the delay on all the links have been doubled for 2X topology. However, for the 5X topology, we observed that the average recovery latency is decreased and is around 0.10 seconds as the packets that would have taken too long to recover were not asked for retransmission. Most of the packets recovered are from receivers present near to each other which has the overall effect of reduced recovery latency for the session.

WAIT Recovery Percentage

In the graph shown in Figure 5.6, we present the effect of link delays on the percentage

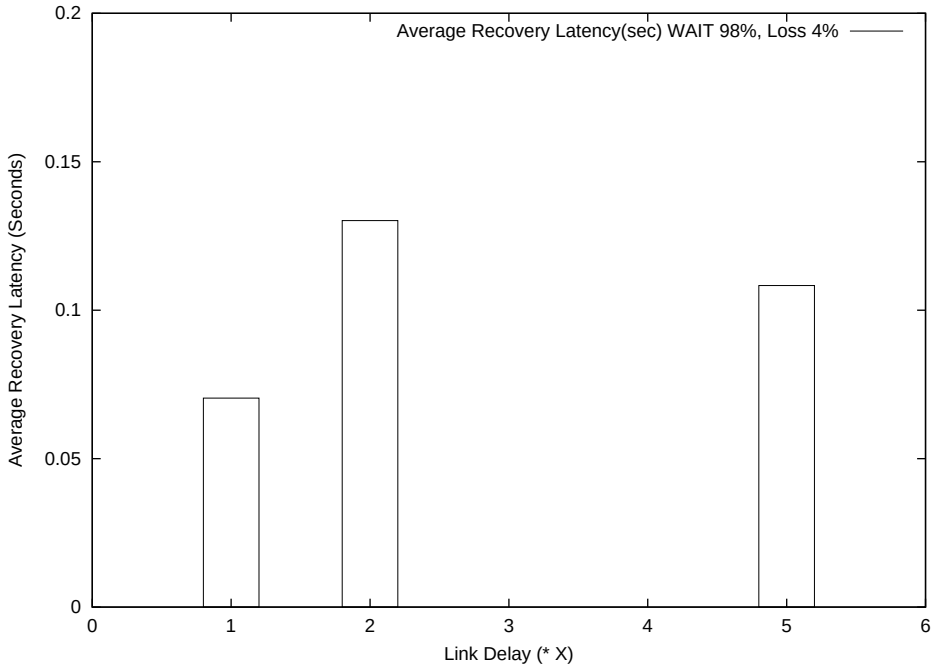

Figure 5.5: WAIT 98% : Recovery Latency *versus* Link Delay for Per-Link Loss of 4%

of packets being recovered. The recovery percentage is calculated by using the ratio of the total number of packets recovered to the total number of packets lost. As seen from the graph, for topology 1X and 2X the recovery percentage is almost same and is around 97.5%. However, for topology 5X the graph has shown that the average recovery percentage is around 92.5% which is as we had expected as some packets have not been recovered because of the higher expected recovery latency.

5.2 Comparison of WAIT, PGM and SRM

In this section we compare the performance of different configurations of the WAIT protocol with PGM and SRM.

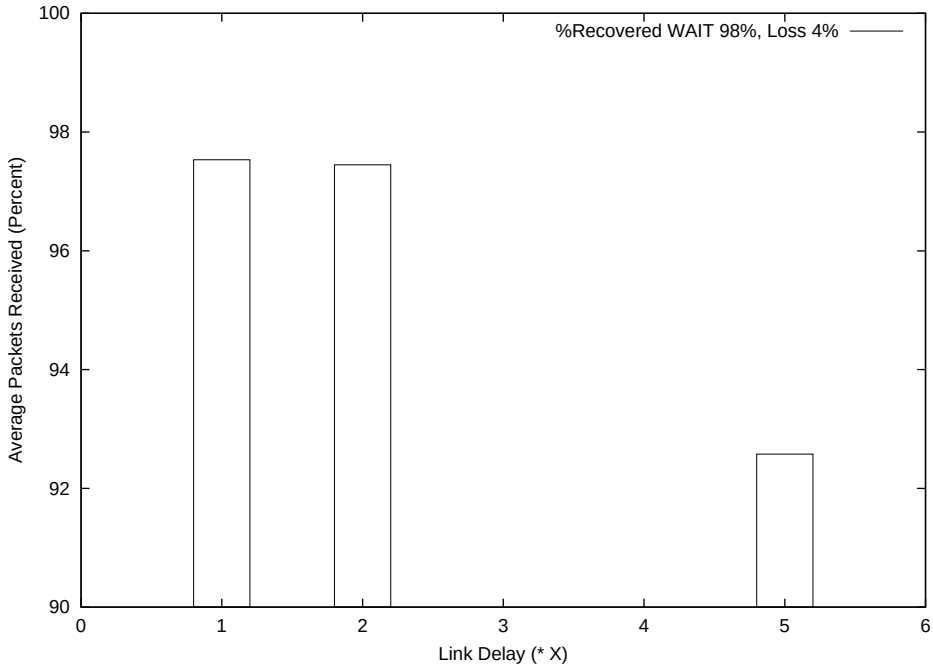

Figure 5.6: WAIT 98% : Average Recovery Percentage *versus* Link Delay for Per-Link Loss of 4%

5.2.1 Performance Comparison: Fixed Link Delays

We compare WAIT with PGM and SRM for the 2X topology and discuss various performance metrics.

Overhead Comparison

The graph in Figure 5.7 compares the three protocols in terms of the average overhead. SRM incurs the most overhead from around 30% for a per-link loss of 2% to around 145% for a per-link loss of 10%. When WAIT is configured at 100% quality requirement, it incurs an overhead from around 10% to around 40% as the per-link loss increases. WAIT 90% shows that there is negligible overhead (0.01%) when the loss per link is 2% as the maximum loss that any receiver can see in this case is 6% as shown in Table 4.2. Since the maximum loss experienced by any receiver is only 6% which is less than the

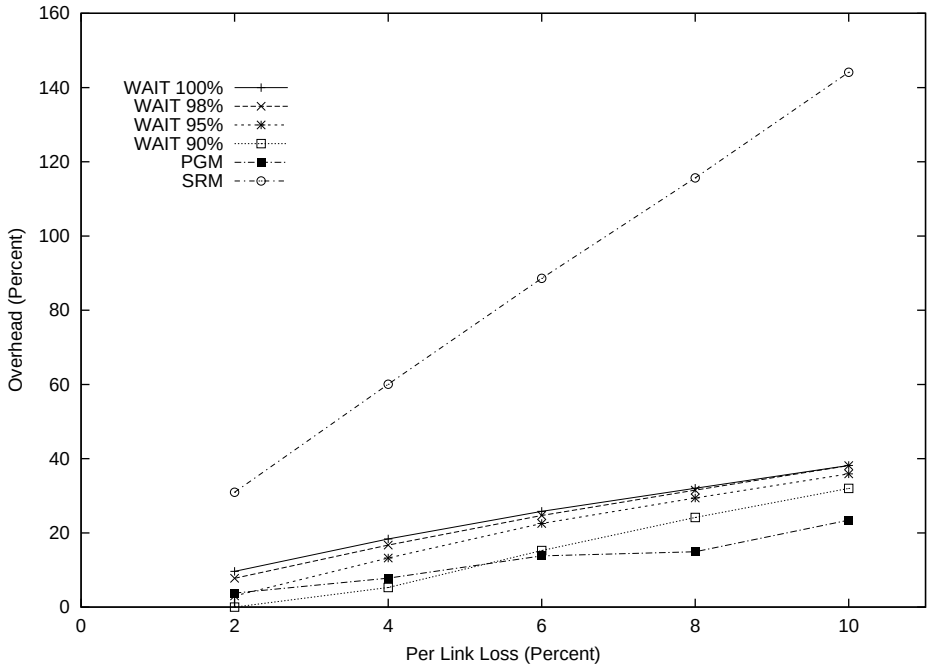

Figure 5.7: Comparison of WAIT, PGM and SRM in terms of the Average Overhead (Percent) with different Per-Link Loss Percentage for the 2X topology.

allowed loss of 10% for the WAIT 90% configuration of the WAIT protocol, the receivers never ask for any retransmissions incurring negligible overhead on the session. PGM shows overhead from 4% to the 24% as the per-link loss percentage increases. Initially, the overhead for WAIT 90% is seen to be lower than PGM until the per-link loss is below around 5.5% but above per-link loss of around 5.5%, WAIT 90% incurs more overhead than PGM, which is a totally router assisted protocol as described in the Section 2.4.3.

End-to-End Latency Comparison

The graph in Figure 5.8 shows the comparison in terms of the average end-to-end latency observed during the simulation runs. As seen from the graph, WAIT and PGM outperform SRM in this area also. As SRM uses the two random back-offs (as described in Section 2.4.1), the average end-to-end latency observed by the SRM session increases from

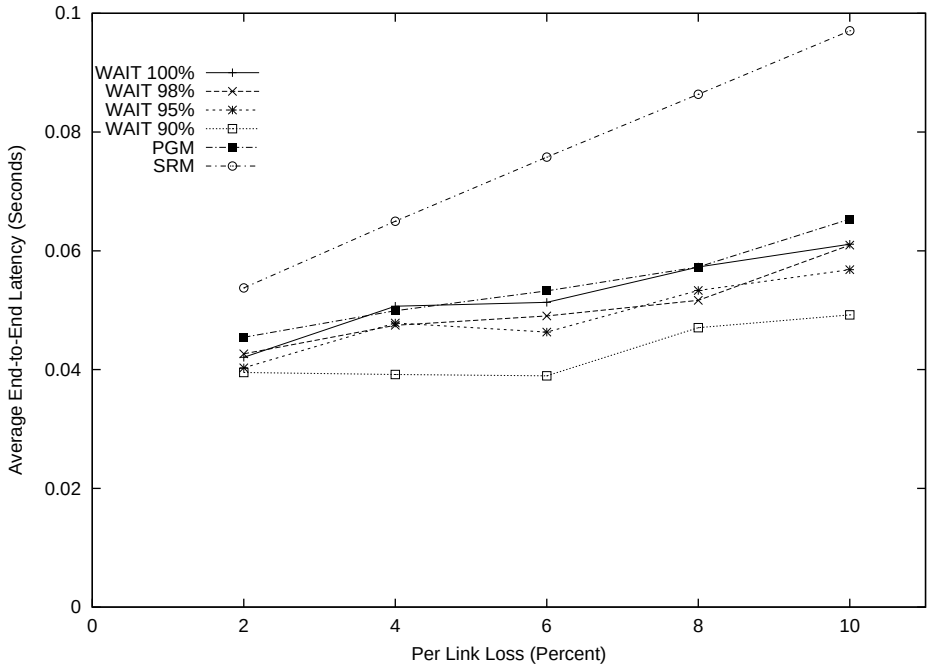

Figure 5.8: Comparison of WAIT, PGM and SRM in terms of the Average End-to-End Latency (Seconds) with different Per-Link Loss Percentage for the 2X topology

around 0.05 seconds to around 0.09 seconds as the per-link loss percentage increases. PGM shows an end-to-end latency of around 0.045 seconds to around 0.065 seconds as the per-link loss increases. WAIT 90% shows that the end-to-end latency remains the same, which is the average end-to-end latency without any repair in the session, to be around 0.041 seconds as long as the per-link loss is less than 6%. All the WAIT configurations have shown the effect of localized recovery as compared to PGM's recovery wherein the request needs to travel to the sender of the session (here the sender might be far away) leading to an increase in the recovery latency. Clearly, the WAIT configurations have shown an improvement in the average end-to-end latency observed by the session compared with the PGM and SRM. So, in effect requesting a repair from a nearby receiver reduces the overall end-to-end latency leading to a better quality for the receivers.

Average Recovery Latency Comparison

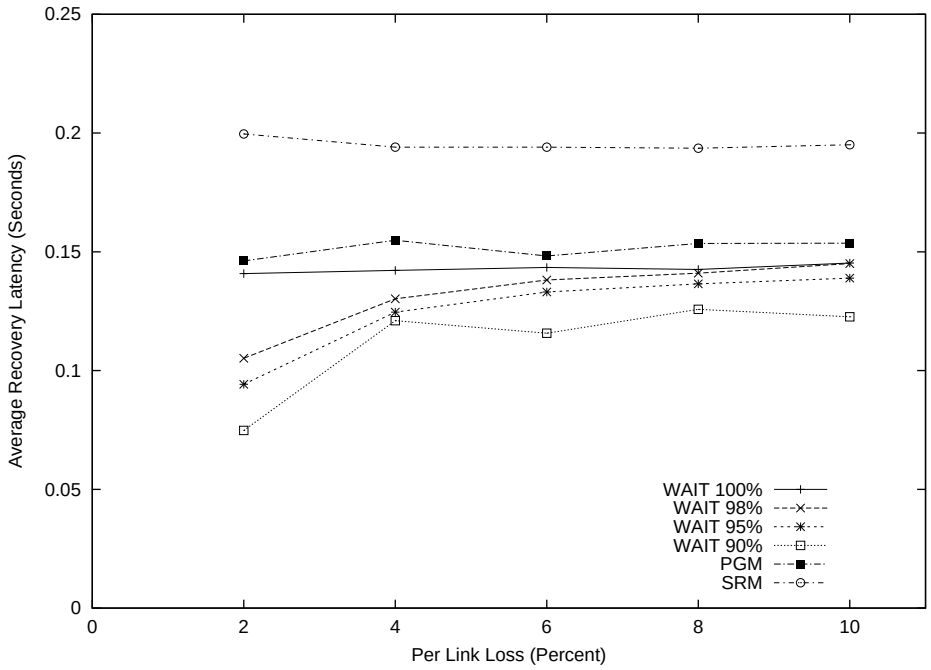

Figure 5.9: Comparison of WAIT, PGM and SRM in terms of the Average Recovery Latency (Seconds) with different Per-Link Loss Percentage for the 2X topology

In the graph shown in Figure 5.9, we observed that SRM shows more recovery latency than both WAIT and PGM. WAIT leads to less average recovery latency because of its local recovery nature as compared to PGM. WAIT 90% at a per-link loss of 2% shows the least recovery latency (0.07 seconds) as the loss occurring on the high delay link (link 1-4) is 2% but as the loss on that link increases, the average recovery latency for the session increases. WAIT 98% and WAIT 95% show a similar trend in the graph. On closely observing the graph for the WAIT 100% and WAIT 98%, we found that for a per-link loss of 2%, there is a large difference between the two values which is the result of some packets that are lost on the high delay link not being asked for retransmission. This also leads to an important observation that combinations of the loss, the link delay and the quality requirement need to be studied in detail to better understand the inter-relationship

between them. This analysis is kept as future work. The graph also provides an insight about how much the buffer size needs to be depending upon the different link delay and quality requirements combinations, which is also an area for further research.

5.2.2 Performance Comparison: Fixed Per-Link Loss Percentage

In this section we compare WAIT, PGM and SRM in terms of Average Overhead, Average End-to-End Latency (Seconds) and Average Recovery Latency (Seconds) for the three topologies 1X, 2X and 5X with maximum and minimum link delays as given in the Table 4.1. Here, we kept the per-link loss to be 8% to carry out the comparison.

Overhead Comparison

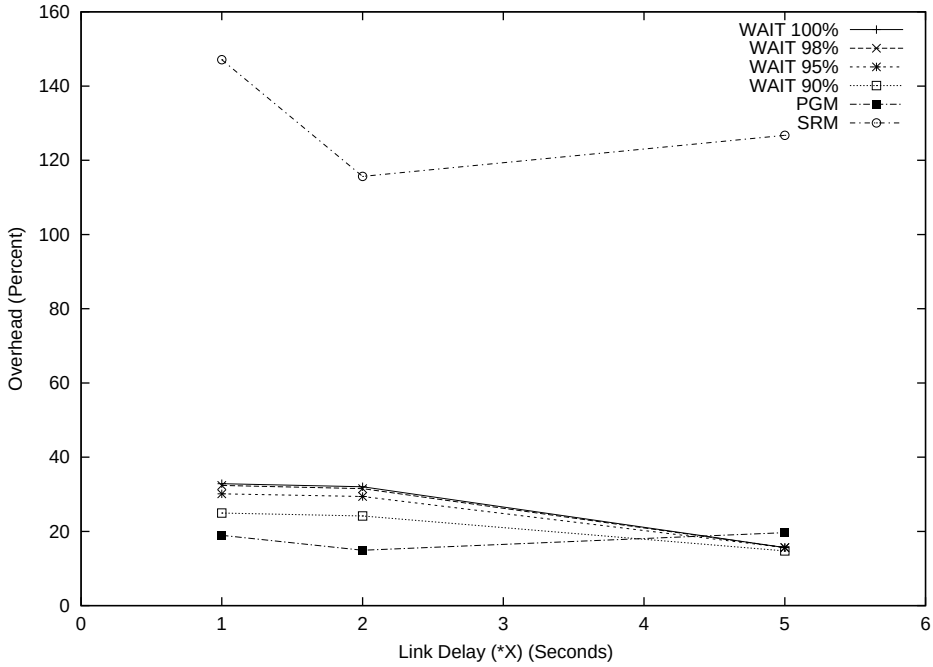

Figure 5.10: Comparison of WAIT, PGM and SRM in terms of the Average Overhead with different Link Delays (Topologies) for the per-link loss of 8%.

In the graph shown in Figure 5.10, we compare WAIT, PGM and SRM in terms of

average overhead incurred during the session using three different topologies of 1X, 2X and 5X link delays. Studying the graph, we observed that SRM incurs more overhead than WAIT and PGM. SRM shows an overhead of almost around 130% overhead for the three topologies. For 1X and 2X topologies, PGM shows a lower overhead (almost around 20%) than either of the WAIT and SRM protocols. For all the WAIT protocol configurations, the overhead remains almost same from around 25% to 35% for the two topologies. However for the 5X topology, all the configurations of the WAIT protocol incur less overhead than the PGM protocol. For 5X topology, the average overhead incurred by the WAIT protocol configurations is around 15% which is almost same for all the configurations. The reason behind this is because of the effect of deciding not to recover losses on the high delay link as described before.

Average End-to-End Latency Comparison

The graph in the Figure 5.11 shows the comparison of WAIT, PGM and SRM for the three topologies (1X, 2X and 5X) in terms of Average End-to-End Latency (Seconds). Again SRM has shown a higher average end-to-end latency than that of WAIT and PGM. For the 1X and 2X topologies the end-to-end latency for PGM and WAIT 100%, WAIT 98% and WAIT 95% are almost same, around 0.030 seconds (for 1X) and 0.05 seconds (for 2x). WAIT 90% shows a still lower average end-to-end latency as compared to the others. However, for the 5X topology, there is big difference between the average end-to-end latency reported by PGM and WAIT. As in case of the high delay links, the WAIT protocol requests fewer packets to be retransmitted as described before, reducing the overall average end-to-end latency. This also shows that requesting the repair from a nearby receiver reduces the overall average end-to-end latency of the session as opposed to PGM's end-to-end latency where request for repair goes all the way towards sender.

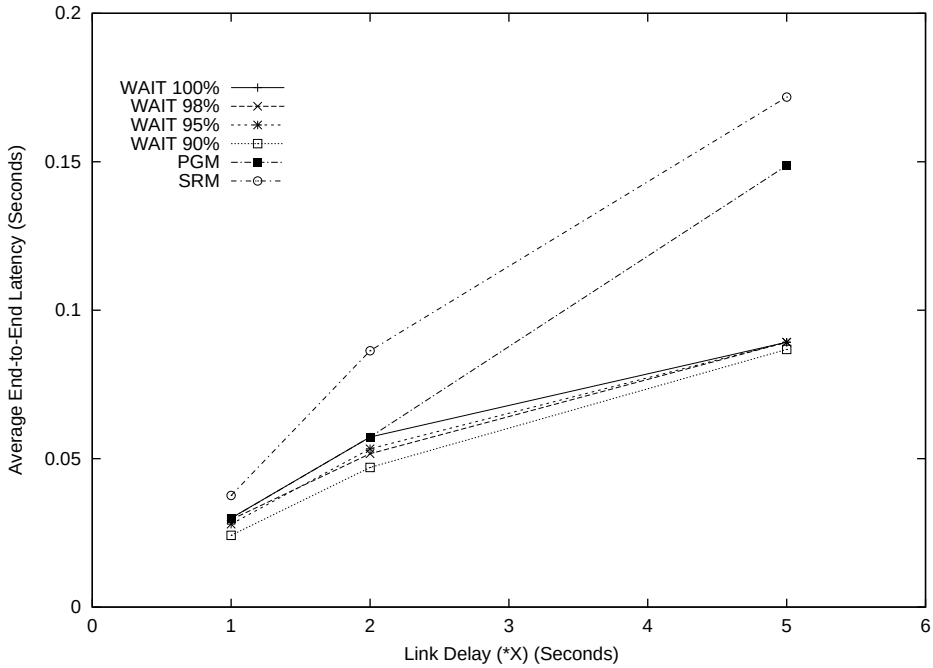

Figure 5.11: Comparison of WAIT, PGM and SRM in terms of the Average End-to-End Latency (Seconds) with different Link Delays (Topologies) for a Per-Link Loss of 8%

Average Recovery Latency Comparison

The graph in Figure 5.12 shows the behavior of WAIT, PGM and SRM protocols in terms of the average recovery latency observed for the three topologies (1X, 2X and 5X). Again SRM shows the effect of two random back-offs before the request and reply transmission as seen from the higher recovery latency in the range of 0.1 seconds to 0.45 seconds. For the 1X topology, PGM and WAIT almost show the same recovery latency in the range of 0.07 seconds. For the 2X topology, PGM has shown somewhat higher average recovery latency (around 0.15 seconds) than WAIT (around 0.12 to 0.14 seconds) as the retransmissions are done from the sender of the session. But for the 5X topology, the average recovery latency for WAIT has shown a decrease (around 0.1 seconds) because of the reason that the loss on the high delay links are not recovered.

We evaluated our approach by comparing it with PGM [8] and SRM [6] in terms of

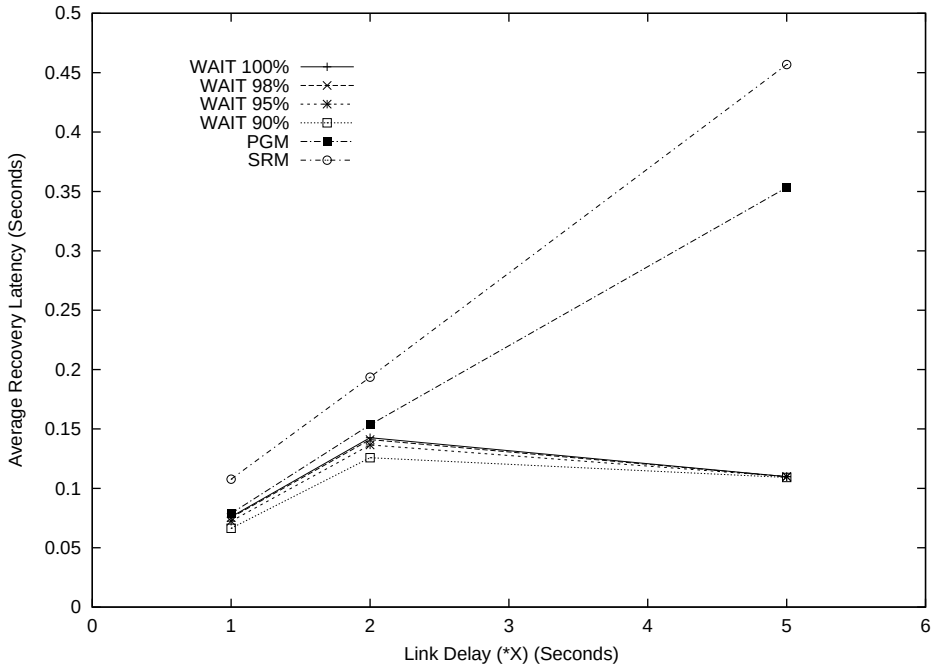

Figure 5.12: Comparison of WAIT, PGM and SRM in terms of the Average Recovery Latency (Seconds) with different Link Delays (Topologies) for a Per-Link Loss of 8%

overhead on the session, average end-to-end latency of the session as well as average recovery latency. We found that the overhead incurred on the session is much less than SRM [6]. We also found that average end-to-end latency for WAIT is also less than SRM [6] as well as PGM [8]. The overhead incurred by WAIT is comparable to PGM which is a totally router assisted protocol. We compared WAIT using different configurations by taking into consideration the continuous media characteristics. For WAIT90% the overhead is seen to be even less than PGM for some scenarios as described in Section 5.2.1. WAIT has shown that it can be tuned to different quality of multimedia sessions. By allowing some loss and also taking into consideration the usefulness of the repair, WAIT has shown that the overhead on network can be reduced to a much lesser extent. By using WAIT for the multimedia multicast applications having different quality requirements we can reduce the overhead on the network as well as improve the recovery latency and end-t-

end latency of the session. Results show that WAIT can be used as a fully reliable as well as resilient multicast protocol which incurs less overhead on the session with improved overall end-to-end latency of the session using localized recovery process.

Chapter 6

Future Work

We have presented and evaluated a new approach for loss recovery during a multicast session over the Internet, which can be used for reliable as well as semi-reliable multicast applications. However, we believe that some more research needs to be done in the direction of our approach for the loss recovery. In this chapter we describe various directions for further research for studying as well as improving our proposed approach.

In our proposed scheme, we have not considered the percentage of loss occurring on the replier link. This loss percentage should be taken into account while selecting a proper local replier as well as upstream replier receiver. In effect, there needs to be a mechanism where the selection of replier receiver will depend upon a proper balance between the loss experienced by the replier receiver as well as the the recovery latency experienced by other receivers because of the receiver being selected as a replier receiver.

While evaluating the WAIT protocol, we have kept the quality requirements for all the participants in a multicast session to be the same. This is normally not the case in the real world and every participant has his own quality requirements during any multicast session(s) based on the play-back buffer size and the allowable loss percentage of each participant. Hence, it would be beneficial to study the behavior of the WAIT protocol

using different quality requirements of the participants.

Research can be done in the area of applying our proposed WAIT protocol for different session sizes. While evaluating our approach we have used a fixed number of participants. Instead, we could study the effect of number of participants in our scheme. In other words, we need to evaluate our protocol in terms of scalability.

More research can be done in the area of studying the effect of play-back buffer size on the recovery percentage and as a result quality maintained by our proposed WAIT protocol.

In our study, we have not evaluated the effect of changing link delays during a multicast session, which is the case in the real world. Additional research that can be carried out is in the area of deciding a proper timeout mechanism which takes into consideration the effects of variable link delays to avoid the implosion as well as exposure problems.

Other research that can be carried out to improve the WAIT protocol is to study the effect of network topology changes on the replier selection as well as on the WAIT and NACK flow. In our study we have not examined the effects of route dynamics as we used a static multicast tree for evaluating our approach.

In our current implementation of the proposed WAIT protocol, we have not evaluated the conditions when NACK, WAIT or REPAIR packets being lost as it would have made the initial analysis of the WAIT protocol difficult to be interpreted. But as now we have shown that WAIT performs well compared to PGM and SRM we intend to study the effects of NACKs, REPAIR or WAIT packets being lost in the network. This is another area where a detailed analysis of the WAIT protocol needs to be carried out.

One more area for studying and configuring the WAIT protocol is that of using content specific treatment to the packets being lost that are asked for retransmission. As an example, for MPEG, more B and P frame losses can be tolerated as compared to I frame losses because of lower quality of the B and P frames than I frames. Hence, the decision

of asking for the retransmission should take into account the importance of the packet for the multimedia application.

In our study, we have done a comparison of our proposed WAIT protocol with PGM [8] and SRM [6]. Another future work would be the comparison study of WAIT protocol with Structure Oriented Resilient Multicast (STORM) [10] protocol.

Chapter 7

Conclusion

Communicating over the Internet using best-effort IP Multicast poses many challenges as discussed in Chapter 1. To recover from the loss of packet(s) in a multicast session a number of approaches have been taken. Continuous media has strict delay requirements, which the sender based retransmission approaches have difficulty taking into account during the loss recovery process. The receiver based loss recovery approaches like SRM [6], PGM [8], ARM [9] provide good solutions to the loss recovery problem but they either trade latency to avoid excess control information and repair traffic or add more functionality in the routers. They also do not take into account that multimedia applications can tolerate some amount of packet loss and hence these solutions recover every lost packet, creating more overhead than required.

In this thesis, we have presented the design and evaluation of our proposed approach, the WAIT protocol, for selective loss recovery for multimedia multicast sessions. WAIT is designed to be tuned for different quality requirements of the multimedia multicast session. WAIT has been designed to reduce overhead on the network as well as improve the quality of multimedia multicast applications over the Internet with less router support. We have done a performance comparison of our proposed WAIT protocol with two other

protocols being developed for the loss recovery of multicast sessions, SRM [6] and PGM [8].

To evaluate WAIT, we developed three agents in NS2 [11] (Wait-Sender, Wait-Receiver and Wait-Agent agents as described in Section 4.1) that carry out the functionality of the WAIT protocol for multimedia multicast applications. We designed the agents in a way that tries to reduce the overhead on routers contrary to router assisted protocols like PGM [8] and ARM [9]. We make the receiver "smarter" by giving them the responsibility of the loss recovery process and avoided having routers to take part in the loss recovery process. We have also implemented a new approach to the trace-route mechanism, *Sender Initiated Trace-Route,* which tries to reduce the number of trace-route packets being injected in the network. To give the session participants some knowledge about the neighboring participants we have also proposed a new *Limited Scope Receiver Information Sub-cast* mechanism.

After implementing the necessary mechanisms in NS2 [11], we evaluated the WAIT protocol for a configuration where we asked the Wait-Receiver agent to maintain the quality for the session of 98% (that is to say we allowed 2% packets to be lost and did not asked for the retransmission of those packet). We found that the maximum average overhead that the WAIT protocol puts on the session is around 40% for a loss of 10% on every link (quite high compared to real world traces). We have also shown that the average recovery percentage done by the agents is almost the same as the quality to which they have been configured.

We also evaluated our WAIT protocol by comparing it with the two other protocols SRM [6] and PGM [8]. We observed that the WAIT protocol performs much better than the SRM protocol in terms of average overhead, average recovery latency as well as the average end-to-end latency. The WAIT protocol has shown results which are almost similar to that of PGM protocol [8] which is a totally router assisted protocol. Some config-

urations of the WAIT protocol agents have shown better results than PGM [8].

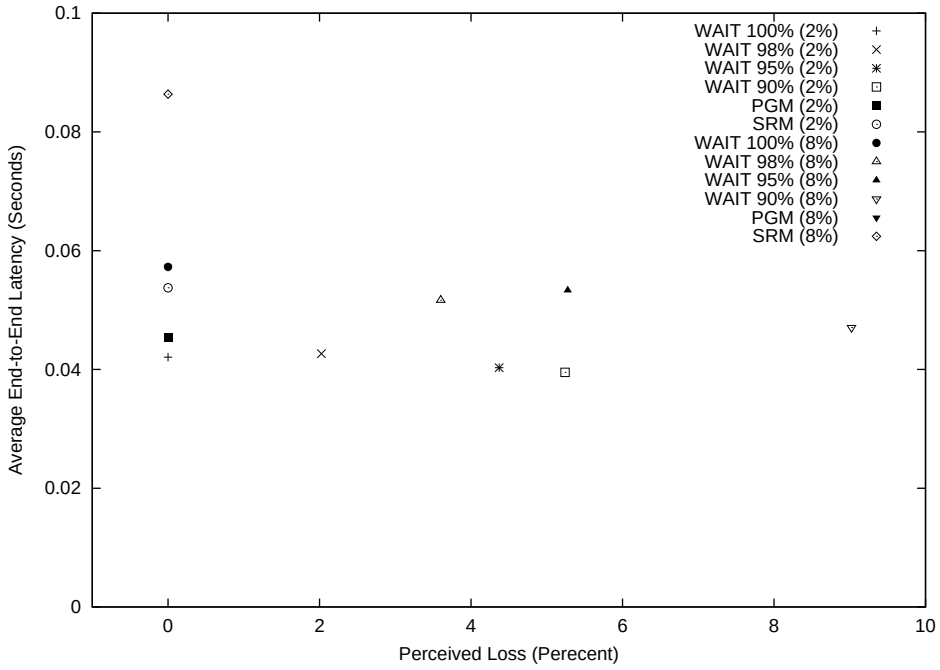

Figure 7.1: Perceived Loss Percentage *versus* Average End-to-End Latency (Seconds)

The graph in Figure 7.1 shows the allowed loss percentage against average end-to-end latency observed by the session for all the configurations of WAIT as well as PGM [8] and SRM [6] agents for the Per-Link Loss Percentage of 2% and 8% (for the 2X topology). Audio applications typically have strict delay and loss requirements. The tolerance of latency for audio applications is around 250 milliseconds for interactive applications. The loss tolerance is around 5% for audio. Hence, it is desirable to stay close to these bounds. The graph shows that if the application requests each and every lost packet to be recovered then the average end-to-end latency of the session is high as shown by the points when the tolerable Loss Percentage is maintained at 0%. However, if we allow for some tolerable loss percentage, the average end-to-end latency of the session has shown to be less. So we can conclude that as multimedia applications can tolerate some amount of packet loss we can get a lower average end-to-end latency as illustrated by the graph in Figure 7.1.

In summary, we designed, implemented and evaluated a new approach for loss recovery for multimedia multicast applications which takes into account the characteristics of the multimedia applications during the loss recovery process. We have also shown that by explicitly notifying (using the WAIT packets) the other receivers of the loss experienced by some receiver(s) we can effectively find whether the loss is a *Global Loss* or *Local Loss* and also the root of the loss subtree. Discovering the root of the loss subtree as well as the type of loss helps us to recover from the loss, effectively reducing the overhead on the network as well as improving the average end-to-end latency of the multimedia multicast session.

Bibliography

[1] S. Deering, “Host Extensions to IP Multicasting,” Tech. Rep. 1112, Internet RFC, Jan. 1989.

[2] M. Yajnik, J. Kurose, and D. Towsley, “Packet Loss Correlation in the Mbone multicast Network,” in *Proceedings of IEEE Global Internet Conference*, Nov. 1996.

[3] M. Handley, “An Examination of Mbone Performance,” Tech. Rep. ISI/RR-97-450, USC/ISI, Apr. 1997.

[4] B. Rajagopalan, “Reliability and Scalability Issues in Multicast Communication,” in *Proceedings of ACM SIGCOMM Conference*, pp. 188 – 198, Sept. 1992.

[5] S. Pingali, D. Towsley, and J. Kurose, “A Comparison of Sender Initiated and Receiver Initiated Reliable Multicast Protocols,” in *Proceedings Of IEEE SIGMETRICS’94 Conference*, 1994.

[6] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang, “A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing,” *IEEE/ACM Transactions on Networking*, vol. 5, pp. 784 – 803, Dec. 1997.

[7] C. Papadopoulous, G. Parulkar, and G. Verghese, “An Error Scheme for Large-Scale Multicast Applications,” in *Proceedings of IEEE INFOCOM’98*, Mar. 1998.

[8] T. Speakman, D. Farinacci, S. Lin, and A. Tweedly, “PGM Reliable Multicast Protocol Specification,” Tech. Rep. draft-speakman-pgm-spec-01.txt, Internet Draft, January 1998.

[9] L. Lehman, S. Garland, and D. Tennenhouse, “Active Reliable Multicast,” in *Proceedings of IEEE INFOCOM’98*, Mar. 1998.

[10] X. Xu, A. Myers, H. Zhang, and R. Yavatkar, “Resilient Multicast Support for Continuous-Media Applications,” in *Intl. Workshop on Network and OS Support for Digital Audio and Video (NOSSDAV97)*, 1997.

[11] U. of California Berkeley, “Network Simulator 2 (NS).” Interent site `http://www-mash.cs.berkeley.edu/ns/`.

[12] W. Fenner, “Internet Group Management Protocol,” Tech. Rep. Internet Draft, v2, Jan. 1997.

[13] T. Ballardie, P. Francis, and J. Crowcroft, “Core Based Trees (CBT) an Architecture for Scalable Multicast Routing,” in *Proceedings of ACM SIGCOMM Conference*, pp. 85 – 95, Sept. 1993.

[14] T. Pusateri, “Distance Vector Multicast Routing Protocol,” Tech. Rep. Internet Draft, Internet Engineering Task Force, Apr. 1997.

[15] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. G. Liu, and L. Wei, “The PIM Architecture for Wide-Area Multicast Routing,” *IEEE/ACM Transactions on Networking*, pp. 153 – 162, Apr. 1996.

[16] J. Moy, “Multicast Extensions to OSPF ,” Tech. Rep. 1584, Internet RFC, Mar. 1994.

[17] U. C. London, “The sdr Session Directory: An MBone Conference Scheduling and Booking System.” Interent site
`http://www.video.ja.net/mice/archive/sdr.html` .

[18] D. Ferrari, “Client Requirements for Real-Time Communication Services,” *IEEE Communications*, vol. 28, no. 11, pp. 65 – 72, 1990.

[19] C. Perkins, O. Hodson, and V. Hardman, “A Survey of Packet-Loss Recovery Techniques for Streaming Audio,” *IEEE Network Magazine*, Sep/Oct 1998.

[20] D. Chiu, S. Hurst, M. Kadansky, and J. Wesley, “TRAM: A Tree-based Reliable Multicast Protocol,” Tech. Rep. TR-98-66, SUN Microsystems Laboratories, July 1998.

[21] D. Clark and D. Tennenhouse, “Architectural Considerations for a New Generation of Protocols,” in *Proceedings of ACM SIGCOMM Conference*, pp. 201 – 208, 1990.

[22] V. Jacobson, “Lightweight Sessions - A new architecture for real-time applications and protocols.” Internet site
`ftp://ftp.ee.lbl.gov/talks/vj-nws93-2.ps.Z`. 3rd Annual Principal Investigators Meeting.

[23] C. Liu, D. Estrin, S. Shenker, and L. Zhang, “Local Error Recovery in SRM: Comparison of two approaches,” *IEEE Transactions on Networking*, vol. 6, Dec. 1998.

[24] S. Paul, K. Sabnani, J. Lin, and S. Bhattacharya, “RMTP: A Reliable Multicast Transport Protocol,” in *Proceedings of IEEE INFOCOM’96*, 1996.

[25] R. Yavatkar, J. Griffioenan, and M. Sudan, “A Reliable Dissemination Protocol for Interactive Collaborative Applications,” in *Proceedings of ACM Multimedia’95*, Nov. 1995.