Router Buffering and Caching Techniques for Multi-Session Reliable Multicast

Qingfeng Xu

A Thesis
in
The Department
of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science at
Concordia University

Montréal, Québec, Canada

April 2003

0-612-77978-5

ABSTRACT

Router Buffering and Caching Techniques for Multi-Session Reliable Multicast

Qingfeng Xu

Reliable multicast has been studied extensively during the past 10 years. Recently, several reliable multicast schemes employing router assistance have been proposed, which not only promise performance gains, but also simplify applications. But the existing schemes based on router assistance didn't consider either simultaneous multiple sessions in network, or the problem of how to share the forwarding buffer of router outgoing port and partition the router cache for multiple multicast sessions so that a better performance of reliable multicast can be achieved. Our work is an attempt to address some of these problems.

In this thesis, we consider a multi-session multicast network and the data caching technique at router for loss recovery. Several policies of router forwarding buffer allocation and router cache partition are introduced for use in the multi-session multicast network. The effect of these policies on the performance of reliable multicast is tested, compared, and analyzed. Additionally, the performance of the ARQ scheme with caching is compared with that of the ARQ scheme without caching. Simulation results show that for the scheme with router caching technique, significant improvements are achieved in terms of end-to-end delay, session transmission time, feedback traffic, and bandwidth utilization.

Acknowledgments

First of all, I would like to express my sincere gratitude to my thesis supervisor, Dr. A. K. Elhakeem, whose guidance, support, and encouragement throughout my work, have made the completion of this thesis possible.

I would also like to thank all of my friends who have ever helped me during research years. It is their generous help that make me overcome many difficulties in my life and study.

Finally, I would like to thank my sister, my brother and my dearest parents. They are always standing besides me and helping me with love, encouragement and trust for all the time.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ACK	ACKnowledgment
AER	Active Error Recovery
ARM	Active Reliable Multicast
ARQ	Automatic Repeat reQuest
BGP	Border Gateway Protocol
BGMP	Border Gateway Multicast Protocol
CBT	Core Based Trees
DMCAST	Directed Multicast
DVMRP	Distance Vector Multicast Routing Protocol
FEC	Forward Error Correction
FIFO	First In First Out
FQ	Fair Queuing
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IP	Internet Protocol
LAN	Local Area Network
LMS	Lightweight Multicast Service
MAC	Medium Access Control
MBGP	Multiprotocol BGP4/Multicast BGP
MBone	Multicast Backbone
MDP	Multicast Dissemination Protocol
MOSPF	Multicast Extensions to OSPF
MSDP	Multicast Source Discovery Protocol
MTP	Multicast Transport Protocol
NAK (NACK)	Negative ACKnowledgement
NCF	NAK Confirmation

ODATA	Original Content Data
OSPF	Open Shortest Path First
OTERS	On-Tree Efficient Recovery using Subcasting
PGM	Pragmatic General Multicast
PIM	Protocol Independent Multicast
PIM-DM	Protocol Independent Multicast Dense Mode
PIM-SM	Protocol Independent Multicast Sparse Mode
QoS	Quality of Service
RDATA	Retransmission Data
RIP	Routing Information Protocol
RM	Reliable Multicast
RMANP	Reliable Multicast Active Network Protocol
RMTP	Reliable MTP
RP	Rendezvous Point
RSC	Reed-Solomon Code
RSE	Reed-Solomon Erasure Code
RTT	Round Trip Time
SPM	Source Path Message
SPT	Shortest Path Tree
SRM	Scalable Reliable Multicast
TCP	Transmission Control Protocol
TP	Turning Point
TTL	Time to Live
UDP	User Datagram Protocol
WAN	Wide Area Network

Chapter 1

Introduction

1.1 Motivation and Scope

With the development of the computer and network techniques, a large number of applications like distributed computing, software updates, distributed caching, etc., which require reliable multicast delivery, are emerging on the Internet. But it is difficult to realize reliable multicast over a best-effort network, especially over Internet. Retransmission request from a large number of receivers can lead to sender and network overload. Additionally, the retransmission from the sender is multicasted to the whole group, even if some receivers don't experience the loss. This will waste network bandwidth and degrade overall performance.

Aimed at the above problems, extensive studies for reliable multicast schemes have been done during the past 10 years. Recently, several reliable multicast schemes employing router assistance have been proposed. These schemes mainly take advantages of network-based processing and storage to control feedback and retransmission problems. In particular, some of these schemes use the routers to cache data for possible retransmission in loss recovery. This approach will efficiently relieve the burden of the source, reduce the network bandwidth for repair traffic, and improve the loss recovery latency. However, these schemes based on router assistance are basically addressed in the study of one multicast group, without considering the condition of simultaneous multiple groups (i.e. multiple sessions). When multi-session data packets exist simultaneously at a router, different forwarding buffer allocation polices for sessions and different cache partition policies for sessions will lead to different performance for reliable multicast. Which policy will be more suitable for reliable multicast? This is the motivation of our research work.

The objectives of this thesis are as follow:

- Studying the performance of reliable multicast for a multi-sessions environment. Given three forwarding buffer allocation policies and three cache partition policies: No-Split, Uniform-Split, and Flexible-Split.
- Ensuring reliability by ARQ.
- Studying the effects of three buffer allocation policies at routers on the performance of reliable multicast.
- Equip some routers with the function of caching data, and evaluate the performance improvement of reliable multicast by comparing with the pure ARQ scheme.
- Study the effects of three cache partition policies at the router on the performance of reliable multicast.
- Vary input parameters, and perform the above studies and comparisons.

1.2 Thesis organizations

In chapter 2, we first introduce IP multicast concepts including the definition, the necessity, and its application, then review some important IP multicast techniques: group management, addressing, and routing. In IP multicast routing, we briefly discuss its two basic routing approaches: dense mode and sparse mode.

In chapter 3, we first review the requirement for reliable multicast, the definition of reliability, and the challenges in design of reliable multicast schemes. Based on the requirement of reliability, two error recovery mechanisms, ARQ and FEC, are described. Furthermore, based on the challenges for reliable multicast, we focus on these reliable multicast mechanisms with router assistance.

In chapter 4, we propose three forwarding buffer allocation policies for multiple sessions, data caching technique for retransmission, and three cache-partition polices for multiple sessions at router, and study the effect of these policies on the performance of multi-session reliable multicast. These policies are to be applied at the router, hence we first give a brief introduction to router architecture. Then we present the simulation models, and give a detailed description of these simulations. Finally, we compare the performance of different policies, and give the performance analysis based on the simulation results.

In chapter 5, we summarize the contributions and conclusions, and suggest the future works.

Chapter 2

IP Multicast

This chapter provides a technical introduction to IP Multicast. Firstly, it gives the definition of IP multicast, discusses the advantage of multicast compared with unicast and broadcast, and reviews the development and application. Furthermore, it introduces the group management, IP multicast addressing, and IP multicast routing.

2.1 Overview

2.1.1 Definition

IP Multicast is described as: "the transmission of an IP datagram to a 'host group', a set of zero or more hosts identified by a single IP destination address" by Steve Deering in [1]. The transmission can be one-to-many or many-to-many, which means one or multiple sources are sending to multiple receivers. The main characteristic of multicast is that the sender only needs to send every datagram once and there is at most one copy of the datagram on every physical link, if the retransmissions do not be considered.

2.1.2 Advantage of IP Multicast

There are three fundamental transmission types for multipoint communication in network [2]: unicast, broadcast, and multicast. Unicast is the transmission of a datagram to a single destination. Broadcast is the transmission of a datagram to an entire subnetwork. Multicast refers to the delivery of datagrams to a set of hosts that have been configured as members of a multicast group across various subnetworks.

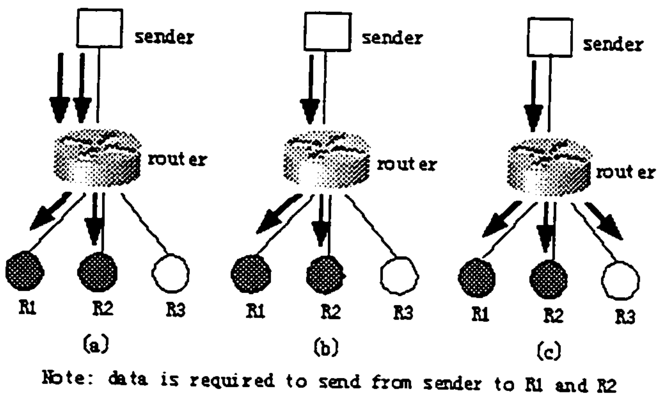


Figure 2.1 Comparison of (a) unicast, (b) broadcast, and (c) multicast

In a point/multipoint to multipoint communication (see Figure 2.1), with unicast, the source will send a copy of the packet to the each group member separately (multiple one to one connection), and it is possible that many copies appear on the same physical link. With broadcast, the packet is forwarded to all outgoing links except the incoming one at each intermediate node, which implies that some hosts receive the packet even if they don't belong to this group. With multicast, just as described in section 2.1.1, the source will send one single copy of the packet for the whole group of members to receive it, and there is at most one copy of the datagram on every physical link along the path from a source to the receivers. From the comparison of the three types of communications, we can easily get that the multicast significantly reduces network traffic, server loads, and bandwidth utilization, so that it is suitable to the point/multipoint to multipoint communication.

2.1.3 Development and Application

Table 2.1 Multicast Applications

	Real-time	Non-real-time
Multimedia	• Video server • Video conferencing • Internet audio • Multimedia events	• Replication: • Video and Web servers • Kiosks • Content delivery • Intranet and Internet
Data-only	• Stock quotes • News feeds • White boarding • Interactive gaming	• Data delivery • Server-server • Server-desktop • Database replication • Software distribution

In 1989, the original model of IP multicast is proposed in [1]. The Mbone, an interconnected set of subnetworks and routers that support the delivery of IP multicast traffic, is built in 1992 [2]. Since then, IP multicast has been tested and implemented on the Mbone (Multicast Backbone). Recently, with increasing the requirement of various multicast applications in the Internet, more multicast research work has been done, and more new protocols and schemes aimed at different issues are proposed. But this is far from being fully developed and there is still huge developing space in this area. With the development of multicast technique, many applications of multicast emerge in multitude in our lives, such that video conferencing, corporate communications, distance learning, distribution of software, stock quotes, news and so on [3], as shown in Table 2.1.

2.2 Groups and Group Management

2.2.1 Concept of Groups

Multicast is based on the concept of a group. In [1], Steve Deering described the group as: "the membership of a host group is dynamic; that is, hosts may join and leave groups at any time. There is no restriction on the location or number of members in a host group. A host may be a member of more than one group at a time." In addition, at the application level, a single group address may have multiple data streams on different port numbers, on different sockets, in one or more applications. Multiple applications may share a single group address on a host.

2.2.2 IGMP

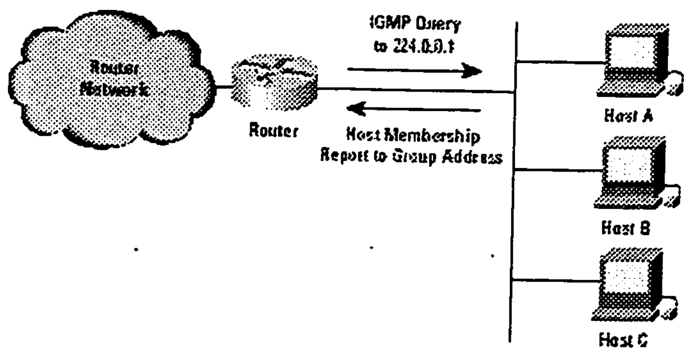


Figure 2.2 IGMPv1 Dialog

Usually, the multicast network uses IGMP to manage groups. The Internet Group management Protocol (IGMP) runs between hosts and their immediately-neighboring multicast routers, as shown in Figure 2.2. It is used to dynamically register individual hosts in a multicast group on a particular LAN. Hosts identify group memberships by sending IGMP messages to their local multicast router. Under IGMP, routers listen to IGMP messages and periodically send out queries to discover which groups are active or inactive on a particular subnet. IGMP uses IP datagrams to carry messages and provides a service used by IP. Therefore, it is thought as an integral part of IP, not a separate protocol [4].

IGMP [1], [5], [6] is the group management protocol currently used in Internet Multicast. Based on the group membership information learned from the IGMP, a router is able to determine which multicast traffic needs to be forwarded to each of its "leaf" subnetworks. Multicast routers use this information, in conjunction with a multicast routing protocol, to support IP multicasting across the Internet.

2.3 IP Multicast Addressing

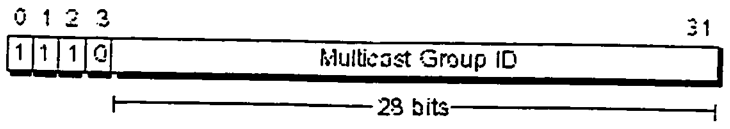


Figure 2.3 Format of Class D IP Address

The range of IP addresses is divided into "classes" based on the high order bits of a 32 bits IP address. IP Multicast uses Class D Internet Protocol addresses to specify multicast host group. The format of the address (IPv4) is shown in Figure 2.3. The first 4 bits, 1110, identify the address as a multicast. The remaining 28 bits specify a particular multicast group. All IP multicast addresses fall in the rage from 224.0.0.0 through 239.255.255.255.

2.4 IP Multicast Routing

Multicast Routing refers to efficiently transmitting multicast datagrams from the source subnetwork(s) to the destination subnetworks. Figure 2.4 shows the family tree of the multicast routing protocol.

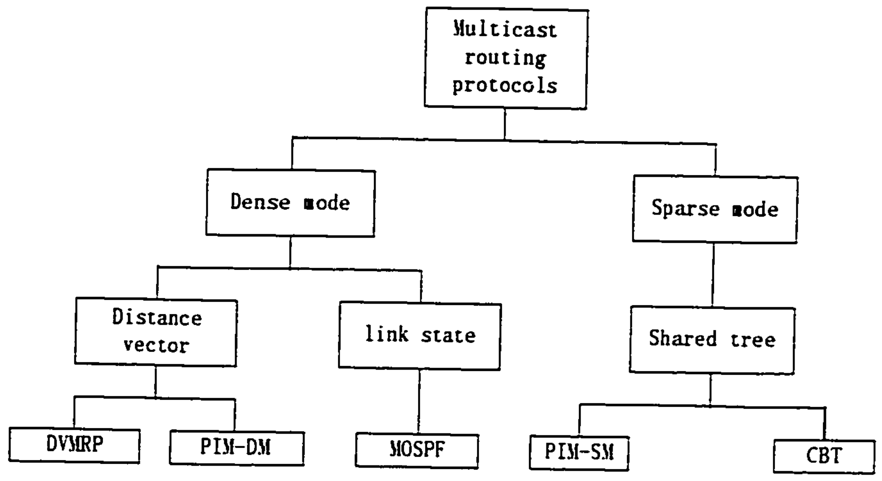


Figure 2.4 Multicast Routing Protocol Family Tree

Multicast routing protocols can generally be classified into two categories, sparse mode and dense mode, which mode is adopted depends on the distribution of multicast group members throughout the network [2]. The Dense-mode is based on the assumption that the multicast group members are densely distributed throughout the network and there is plentiful bandwidth, while the Sparse-mode is based on the assumption that the multicast group members are sparsely and widely distributed throughout the network, for example across many regions of the Intemet. We will review the two modes protocols hereinafter.

Note the protocols in Figure 2.4 are intradomain protocols. To perform multicast between different autonomous administrative entities, the interdomain protocols are required, such as MBGP [7], MSDP [8] and BGMP [9]. In this thesis, we won't discuss them in detail.

2.4.1 Multicast Distribution Trees

In order to better understand the multicast routing algorithms, we first introduce multicast distribution trees.

Multicast enable routers use multicast distribution trees to control the path which IP Multicast traffic takes through the network in order to deliver traffic to all receivers. There are two basic types of multicast distribution trees: source trees and shared trees [2].

- **Source Trees** [2]

The simplest form of a multicast distribution tree is a source tree with its root at the source and branches forming a spanning tree through the network to the receivers. Multicast routing protocols using this algorithm will build a tree for each source or each source/group pair. Because this tree uses the shortest path through the network, it is also referred to as a shortest path tree (SPT). The example of source tree is shown in Figure 2.5.

Shortest Path Trees have the advantage of creating the optimal path between the source and the receivers. This will guarantee the minimum amount of network latency for forwarding multicast traffic. However, the routers must maintain path information for each source. In a network that has thousands of sources and thousands of groups, this can quickly become a resource issue on the routers, and limit the scalability of applications with many active senders. Hence, memory consumption due to the size of the multicast routing table is a factor that network designers must take into consideration.

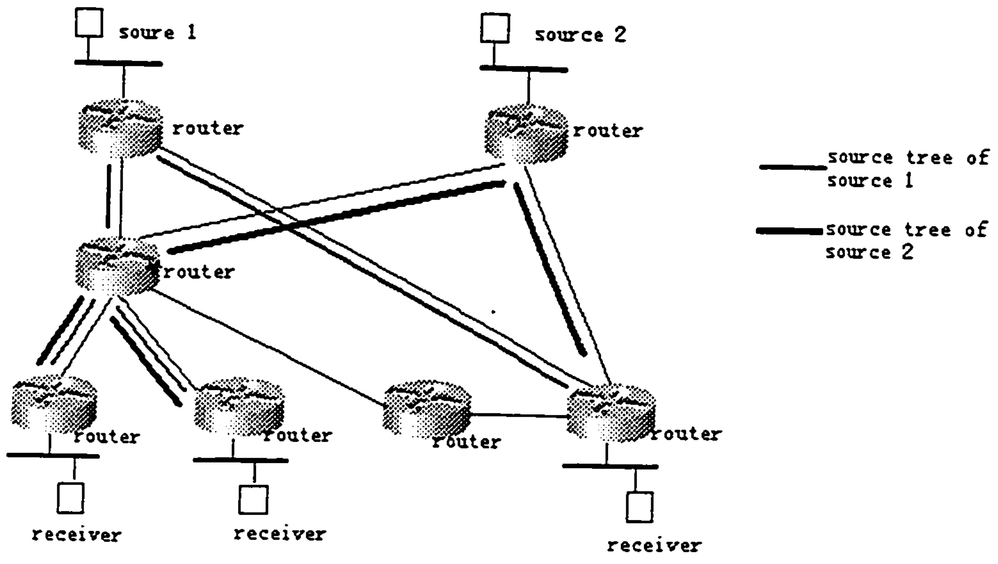


Figure 2.5 Source Trees

- **Shared Trees** [2]

Unlike source trees that have their root at the source, shared trees use a single common root placed at some chosen point in the network. This shared root is called a Rendezvous Point (RP). The protocols using this algorithm construct the delivery tree(s) that is shared by all members of a group. The example of shared tree is shown in Figure 2.6.

Shared Trees have the advantage of requiring the minimum amount of state in each router. This will lower the overall memory requirements for a network that only allows shared trees. The disadvantage of shared trees is that under certain circumstances the paths between the source and receivers might not be the optimal paths—which might introduce some latency in packet delivery. In addition, shared trees may result in traffic concentration and bottlenecks near core routers, hence it must be carefully considered about the placement of the RP in a shared tree only environment.

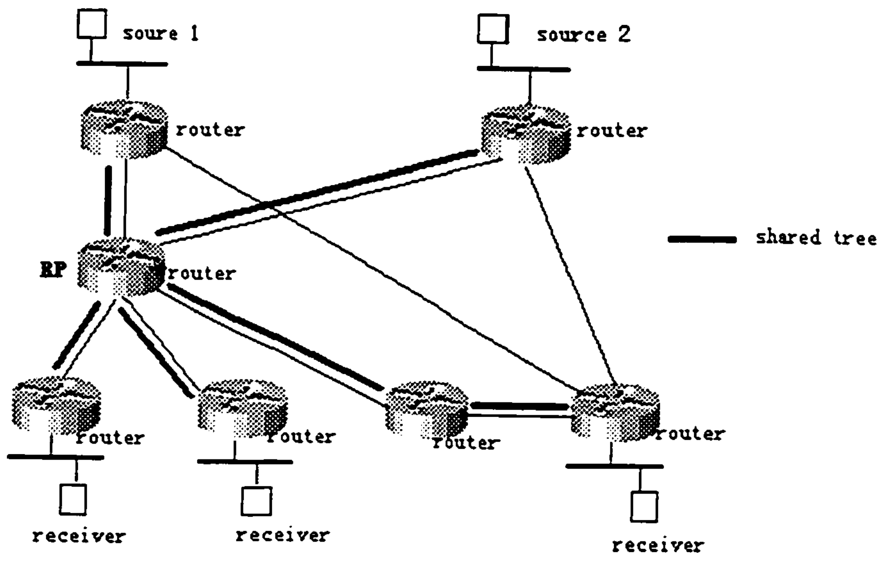


Figure 2.6 Shared Tree

2.4.2 Dense Mode Protocols (DVMRP, MOSPF, PIM-DM)

Dense mode protocols use source-based shortest path tree algorithms. To inform other routers of multicast sources, this mode protocol uses a push model to flood the multicast traffic to all routers in the network. A router with no receivers interested in this traffic will then tell its upstream router to stop forwarding this traffic or to prune this branch from the tree. This flood-and-prune mechanism allows these protocols to easily build a multicast distribution tree rooted at the source. A source-based tree guarantees the shortest and most efficient path from source to receiver. While this may be an ideal enterprise solution in many circumstances, the reliance on broadcast and flooding across the Intemet simply will not scale.

Examples of dense mode protocols are Distance Vector Multicast Routing Protocol (DVMRP) [10], Multicast Open Shortest Path First (MOSPF) [11] and Protocol Independent Multicast Dense Mode (PIM-DM) [12]. DVMRP protocol incorporates the distance vector algorithm to provide routing information, while MOSPF depends on the link-state routing protocol OSPF version2 [13]. Unlike the above two protocols based on a specific unicast routing algorithm to provide routing information, PIM does not depend on a certain underlying unicast routing protocol, which could use any underlying unicast routing protocol to build the multicast distribution tree.

2.4.3 Sparse Mode Protocols (CBT, PIM-SM)

Sparse mode multicast routing protocols use shared tree algorithms, which have a better scalability than dense mode protocols and are best suited for the environment with widely dispersed group members in a wide area network.

This mode protocol uses a pull model to deliver multicast traffic. Only networks that have active receivers that have explicitly requested the data will be forwarded the traffic. When a source begins to actively send multicast traffic, its directly connected router, or designated router, registers with the RP. The RP will keep track of all active sources in a domain. When a router is connected to a host that wants to receive a multicast group, it will use RPFs to determine the shortest path to the RP. While the RP builds a tree to the source, all receivers join the tree at the RP. The multicast traffic will be forwarded to the receivers along the shared tree. As long as all routers know which router is the RP, broadcast is not needed to distribute multicast routing information.

Protocol Independent Multicast Sparse Mode (PIM-SM) [14] and Core-based tree (CBT) [15] are examples of sparse mode routing protocols. There are three main differences between PIM-SM and CBT. The shared trees built in CBT are bi-directional, while in PIM_SM they are uni-directional; In addition, PIM-SM trees are "soft state", maintained by periodical "join" message, while CBT trees are "hard state" and an explicit tear down message is needed to delete a state; Finally, if the traffic volume exceeds a certain threshold, in PIM-SM a router can switch from the shared tree to a shortest path tree.

Chapter 3

Reliable Multicast

Multicast has become an important component of the Internet within the past decade, due to its advantages discussed in Chapter 2. Many multicast applications require reliable delivery. But the network layer multicast [2] only offers best-effort one-to-many or many-to-many delivery service and offers no guarantees. Therefore, reliable multicast transport protocols on top of IP multicast are required to guarantee reliable delivery.

Designing a generalized "One-size-fits-all" multicast transport protocol is a difficult task [16], hence a large number of reliable multicast protocols are proposed to solve the different needs of particular applications. But generally, the error handling mechanisms of these protocols can be classified into Automatic Repeat reQuest (ARQ) and Forward Error Correction (FEC). A recent trend in reliable multicast has been focusing on adding router support for fast and efficient loss recovery, and these schemes with router assistance are proved to be able to achieve significantly better performance than non-assisted schemes. These protocols and schemes will be described in detail.

This chapter will first give the definition of the reliability. Next it will review the challenges, problems, and the solutions in design of reliable multicast protocols. Finally, these ARQ/FEC mechanisms and router supporting schemes are introduced in later sections.

3.1 Definition of Reliability

There are different definitions to "reliability", depending on different applications. In general, the broad-sense "reliability" includes error-free delivery [17], atomicity [18], and ordering [18]. The first property, error-free delivery refers to delivering all data to all receivers eventually, which will be further discussed hereafter. The second property, atomicity, guarantees either all of the applications/processors or none of them receive a message. Ordering refers to keeping the time precedence relations between multicast messages. In this thesis, we will focus on the first property.

Error-free delivery, is the narrow sense "reliability" used in Internet multicast, which can be further classified into semireliability which means a certain level of errors can be tolerant, time-bounded reliability which means strict delay and/or delay jitter bounds are required, and full reliability which indicates delivery of all data to all receivers is required to be fully error-free.

To better understand these concepts of the narrow sense "reliability", we will give some application examples about them. Bulk-data transfers, such as file distribution and web cache updates, are both fully reliable and not time-bounded. On the other hand, some applications require not only full reliability but also time-bounded delivery, such as shared whiteboards and distributed games. Furthermore, real-time streaming applications like video/audio multicasting interactive applications like video conferencing are both semi-reliable and time-bounded.

3.2 Challenges Facing Reliable Multicast

There are many challenges in reliable multicast. This section introduces two main problems: scalability and congestion control.

3.2.1 Challenge I – Scalability for Loss Recovery

Scalability indicates the ability to have large groups for a reliable multicast scheme. It is a fundamental issue to a successful reliable multicast protocol [19]. There are two main issues related to scaling: feedback implosion and retransmission exposure.

- **Feedback Implosion**

ARQ technique is usually used in reliable transport of network. For multicast, a large amount of feedback traffic (NAK or ACK), which is sent synchronously by many receivers, can result in network congestion and overwhelm the sender. This is so-called feedback implosion [20]. Basically, there are three solutions for this problem.

The first approach is timer-based NAK suppression at the receivers. Each receiver will set up a timer for a NAK and send it at timeout. If loss has been corrected or the receiver has received a NAK containing the same information, the timer is reset and the NAK is cancelled. In [21], the scheme based on this approach is proposed.

The second approach is hierarchical ACKs aggregation and/or NAK suppression at some intelligent routers/servers. This approach uses a tree structure. RMTP-II [22] is such mechanism.

The third approach is using proactive FEC (forward error correction). Send some parity data packets with the source data packets. When loss is detected at receivers, the receivers can use the redundancy information to correct the loss. In [23], this mechanism is adopted.

- **Retransmission Exposure**

Retransmission exposure means that recovery-related traffic reaches the receivers which have not experienced any loss. In reliable multicast, usually, the sender is the ultimate responder of retransmission request and it retransmits repair packets by multicast. Hence, when group size is very large, if the sender responds to every retransmission request, this approach possibly leads to overload of the source, a relatively large recovery delay, and retransmission exposure. Retransmission exposure reduces the network bandwidth efficiency. Especially when only a small fraction of the receivers suffer loss persistently to request retransmissions (the "crying baby problem") [19], this problem is worse.

To solve this problem, several schemes can be adopted. Hybrid FEC/ARQ technique can partly alleviate the effect [23]. Another solution is using hierarchical structure, distributing the burden of retransmissions among the sender and some retransmission servers (receiver, router, or server) and limiting the scope of retransmission [21].

3.2.2 Challenge II – Congestion Control

Multicast congestion control is set of techniques that regulate the data transmission rate in response to network conditions and the principles and mechanisms of sharing congested links among many sessions. The primary goal for multicast congestion control is avoiding congestion collapse and achieving fairness with competing traffics to utilize the network resources efficiently. The major issues for multicast congestion control are scalability, heterogeneity, and fairness. Multicast congestion control is more complicated than unicast congestion control.

- **Scalability for Congestion Control**

A multicast congestion control protocol not only needs to scale to a large number of receivers but also needs to scale in a more heterogeneous environment with different link capacities and delays. This can results in two problems. One is aforementioned feedback implosion, which won't be repeated. The other is the loss path multiplicity (or rate drop-to-zero) problem [24]. The loss path multiplicity problem arises when receivers use packet losses as congestion signals and the source uses these signals to regulate its transmission rate without proper aggregation. When packets are lost on multiple paths independently, receivers downstream of these paths will all send congestion signals to the source resulting in multiple rate drops at the source. Decoupling feedback for error control and feedback for congestion control can solve this problem. [25].

- **Heterogeneity**

Another major issue for multicast congestion control is the heterogeneity of group members and network capacities. For example, the bottleneck link capacity leading to receivers in a multicast group can vary from 33.6 kb/s for a dialup link to more than 100 Mbps in a LAN. It is often desirable for a receiver to have a transmission rate that matches its receiving rate. This leads to the inter-receiver fairness requirement: the transmission rate of a multicast group should satisfy faster receivers in the group while not overwhelming slower ones at the same time. The multi-rate multicast congestion control schemes [26] can help improve inter-receiver fairness.

- **Fairness**

Fairness in multicast congestion control mechanisms refers to sharing the network resources equally among receivers or sessions, which includes inter-receiver fairness and inter-session fairness. Inter-receiver fairness means the fairness among receivers in a multicast session, which has been mentioned above. Inter-session fairness refers to fairness among multicast sessions and between multicast and unicast sessions (such as TCP flows).

The definition of "intersession fairness" is largely policy-based. Different definitions are possible due to various requirements of applications, customers, and service providers. Popular fairness criteria include max–min fairness [27] and "TCP-friendliness" [28].

3.3 Error Recovery Mechanisms in Reliable Multicast

IP multicast provides only unreliable and best effort delivery at the network layer. To deal with loss, two basic mechanisms exist: ARQ (Automatic Repeat Request) which retransmits the lost data, and FEC (Forwarding Error Correction) which transmits redundant data including the original data and the parity data. If the amount of original data lost is not more than the amount parity data sent, the parity data can be used to reconstruct the lost original data.

3.3.1 ARQ

ARQ [29] can achieve reliable delivery by the receiver explicitly (via NAKs) or implicitly (via ACKs and timeouts) requesting retransmission of the lost segments. The ARQ mechanisms include three parts. The first part is loss detection, which can be done by the receiver (gap-based loss detection or timeout) or by the sender (timeout). The second part is sending feedback, either positive acknowledgements (ACKs) or negative acknowledgements (NAKs). The last part is the lost data retransmission, which is the responsibility of the sender or other nodes (receiver, router, etc.). There are two main retransmission schemes, Go-back-N and Selective, which trade off simplicity of the receiver implementation and transmission efficiency.

ARQ based reliable multicast protocols can be classified into sender-initiated and receiver-initiated protocols, according to their different feedback mechanisms [30]. In sender-initiated reliable multicast protocols based on the use of ACKs [30], the sender maintains state information of all receivers and detects packet losses. Receivers need to acknowledge every correctly received packet via ACK to the sender. If the sender does not receive the ACK for a packet after time out, it will assume that the packet is lost and a retransmission will be triggered. In receiver-initiated reliable multicast protocols based on the use of NAKs [30], most of the responsibility for reliable data delivery is shifted to the receivers. Each receiver is responsible for detecting loss by observing gaps in received packets and informing the sender via NAKs when it requires the retransmission of a packet.

Receiver-initiated protocols have the advantage of being scalable to a large number of receivers than sender-initiated protocols [30], since the receiver-initiated protocols distribute the burden of maintaining reliability into all receivers, NAKs are only issued when packet losses occur, and the sender need not keep the state or identity of each receiver. But these two types of protocols will be confronted with the same problems for scalability, how to alleviate feedback explosion, how to resolve retransmission exposure, and how to provide timely delivery. Aimed at these problems, many protocols with hierarchical mechanism based on tree structure, are proposed, such as RMTP [17] and RMTP-II [22].

However, the pure ARQ mechanisms scale badly to multicast protocols and large groups, because they would generate excessive traffic and high network latency due to its feedback transmission and each lost data retransmission. FEC mechanisms have thus been introduced for reliable multicast.

3.3.2 FEC

3.3.2.1 Introduction to FEC

FEC [31], forward error correction, is an error detection and correction technique based on transmitting redundant information, which reconstructs some amount of missing data by using the parity packets instead of the retransmission of the lost data. The different properties on mechanism between FEC and ARQ, decides the differences of their application scenarios in reliable mulitcast, as shown in Table 3.1.

Generally, FEC has the following benefits for reliable muliticast: Firstly, it has a faster recovery of missing packets and a lower latency to receive all data intact at all receivers, compared with ARQ recovery with a larger RTT (including the time of sending NAK and the time of lost data retransmission). In addition, FEC can improve scalability in terms of group size, because it can significantly reduce the necessity for retransmission request or make them totally unnecessary by large improvements in the packet loss rate. Furthermore, FEC can improve transmission efficiency for reliable multicast, since different receivers with different loss patterns can be recovered using the same set of transmitted data. Finally, FEC has been implemented in software without an excessive overhead [32], which makes it possible to widely use FEC.

However, FEC by itself cannot provide full reliability, because the sender does not receive any feedback from the receivers about their losses, thus there is no way for the sender to know how much redundancy is needed to fully recover lost data. Therefore, merging ARQ and FEC is necessary for reliable multicast protocols, which will be discussed later.

Table 3.1 Different Application Scenarios for Pure FEC and ARQ

FEC	ARQ
Suitable for large groups with large Round-Trip Times(RTTs), or when the feedback channel is unavailable	Suitable for small groups with feedback channel
Suitable for networks with homogeneous loss probability	Suitable for networks with heterogeneous loss probability
Efficient in overcoming independent loss	Efficient in overcoming shared loss
Suitable for real-time interactive applications	Suitable for non-interactive applications
Only provides semi-reliability	Provides total reliability

3.3.2.2 Erasure Correction Codes

FEC corrects loss or error by using FEC codes. Generally speaking, FEC correction includes both erasures (which is usually generated by network congestion) and bit-level corruption (which is usually generated by line noise) corrections. However, in computer network, the link layer or transport layer can use packet authentication to discard corrupted packets, which leads to the transform from unrecoverable corruption to erasure [32]. Hence, transport layer FEC will only be required to deal with the packet-level erasures by using erasure correction codes.

Erasure codes are a subset of Error Control Codes, largely used in the telecommunication field [32]. Basically, a *(n, k)* block erasure code takes *k* source packets with same packet length and produces *n* encoded packets that include k source packets and n-k parity packets with the same packet length. Any subset of *k'* (for RSE k'=k; for Tornado code *k'*>*k*) encoded packets allows the reconstruction of the source packets, such as shown in Figure 3.1 [32].

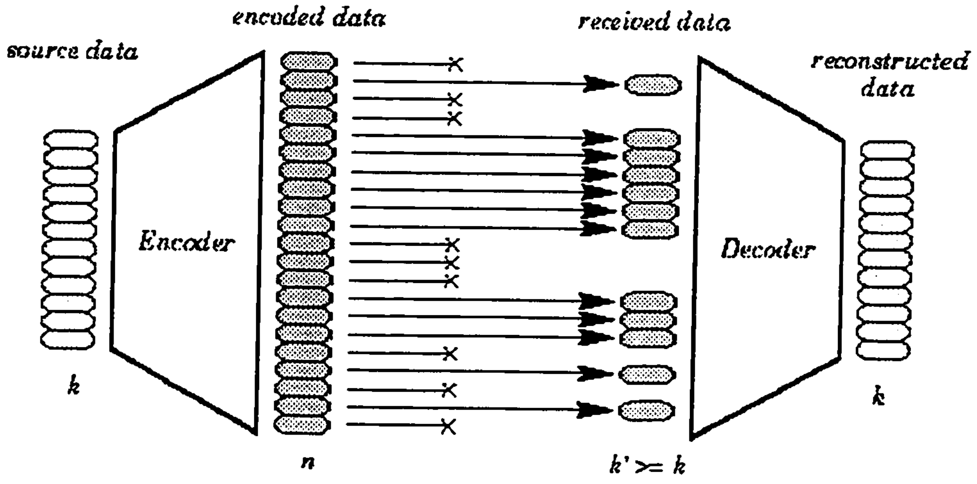


Figure 3.1 Decoding and Encoding Process of *(n, k)* Block Erasure Codes

The following subsections describe the commonly employed coding schemes for reliable multicast: Reed Solomon Erasure (RSE) code and Tornado code.

- **RSE Code**

The Reed-Solomon erasure correction code (RSE) [31] is a small block FEC code, which is derived from the well-known Reed-Solomon error correction code (RSC). The major difference between RSE and RSC is that RSE only corrects erasures. RSE is used in many FEC-based reliable multicast protocols.

RSE code can be constructed based on the properties of linear algebra over finite fields [32]. It interprets k source packets as the coefficients of a polynomial P of degree k-1. As the polynomial is fully characterized by its values in k different points, we can produce the desired amount of redundancy by evaluating P at n different points (i.e. n different finite fields elements). Reconstruction of the original packets (the coefficients of P) is possible as soon as any k of these values are available. In practice, the encoding process requires multiplying the original packets by an $n \times k$ encoding matrix G, which happens to be a Vandermonde matrix. The decoding process requires the inversion of a $k \times k$ encoding submatrix G' taken from G, and the multiplication of the received packets by G'^{-1} (inversion matrix of G'). By simple algebraic manipulation, G can be transformed to make its top k rows constitute the identity matrix, thus making the code a systematic code.

The Reed-Solomon code [31][32] is computationally expensive. The encoding and decoding times of RSE are relatively high, and increase as the size of the original data block increases. Usually, the data block size has to be kept as small as possible. While hardware implementation is faster than software implementation, hardware FEC is more common in telecommunication equipment rather than in PCs and workstations for end-to-end transport protocols. However, Rizzo et al. [32] showed that software FEC is becoming feasible as the CPU speed increases. For k values under 32, his software implementation can encode at 10 megabytes/second and decode at 9 megabytes/second on even a low-end PC.

- **The Tornado Code**

Tornado code [33] is a large block FEC code that provides an alternative to small block FEC codes. The mathematical basis of the Tornado code is also linear algebra, which produces linear-time encodable and decodable codes based on a series of random bipartite graphs. To ensure that a receiver can reassemble efficiently the object with low reception overhead, the packets are permuted into a random order before transmission. The decoding process can be achieved using only simple substitutions and XOR operations. When close to required number of packets are received, the arrival of one more packet may trigger a whirlwind of substitutions and regenerate a number of original packets. Hence this code is named "Tornado". The technique is described in detail in [33].

Tornado code [33] promises fast encoding and decoding, and overcomes the limitation of small block size of RSE. The advantage of Tornado code over RSE is that it can trade off a negligible increase in reception overhead for a substantial decrease in encoding and decoding times. That is, a (n, k) Tornado code requires slightly more than k out of n encoding symbols to recover k source symbols, i.e., there is a small reception overhead, while the Reed-Solomon code requires only exactly k out of n encoding symbols received in order to reconstruct the original k symbols. In return, the Tornado code encoding and decoding involve only addition (i.e., XOR) and no matrix inversion (which requires more costly multiplication, typically using table lookups).

3.3.3 Hybrid FEC/ARQ

In reliable multicast schemes, FEC is often used together with ARQ, called hybrid FEC/ARQ [29][34]. In hybrid FEC/ARQ schemes, instead of retransmitting individual lost packets in pure ARQ schemes, the source transmits parity packets either proactively with the original data or in response to a NAK (or ACK timeout). Receivers can use the parity packets to recover any losses in a range of data packets. Hybrid FEC/ARQ techniques combine the benefits of FEC and ARQ, and are significantly suitable for reliable multicast, hence so far a large number of reliable multicast protocols with hybrid FEC/ARQ are proposed.

Hybrid FEC/ARQ techniques are classified into two categories: hybrid FEC/ARQ I (proactive FEC) and hybrid FEC/ARQ II (reactive FEC), depending on how to use FEC [29][34].

3.3.3.1 Proactive FEC

In proactive FEC, for each block, the source transmits both original packets and extra parity packets, regardless of the receiver's feedback. Retransmissions are still necessary to ensure reliability but are fewer because proactive FEC effectively reduces the end-to-end loss rate. The approach makes more efficient use of network resources than an approach with pure ARQ when losses are not temporally very burst [35].

But for proactive FEC, it is difficult to decide the appropriate amount of parity packets to send proactively for source in heterogeneous network. Any amount may be insufficient to some receivers while unnecessary to others [35]. A scheme, which intends to resolve the above problem by estimation of loss rate based on feedback and localization traffic, was presented in SHARQFEC [36].

3.3.3.2 Reactive FEC

Reactive FEC, i.e., hybrid FEC/ARQ II, was proposed in [37]. Reactive FEC more efficiently utilizes the network resources than proactive FEC. It can reduce the usage of network resources, even when losses are temporally correlated. In this approach, FEC packets are transmitted only as retransmissions, either upon timeout waiting for an ACK or in response to a NAK. Reactive FEC has the following properties:

- Retransmissions are no longer the lost source packets of certain receiver but parity packets that can help recover any lost packets.
- The same parity packet can be used to repair the loss of different data packets at different receivers.
- A NAK from a receiver only simply indicates the number of the lost source packets in certain block instead of the sequence number of each lost source packet.
- Source picks the maximum number of losses reported by all the NAKs for the same block and multicasts that amount of parity packets to the multicast group.

However, because the source has to retransmit according to worst case and the responsibility of retransmission is centralized at the source [35], the schemes exist the problems of crying baby [19] and NAK implosion [20](mentioned in section 3.2), although FEC reduces the number of NAKs. Of course, we can ease the problem by localizing the retransmission into subgroups [36]. But the fewer the receivers that share the repair packet (due to localization), the less FEC improves the performance over normal ARQ. Therefore, reactive FEC is more suitable for a large set of homogeneous (in terms of loss rate) receivers with some form of NAK suppression such as duplicate avoidance or aggregation along the fusion tree.

3.4 Router Support for Reliable Multicast

3.4.1 Background

In the past ten years, many solutions have been proposed on a high layer (transfer layer and application layer) for reliable multicast, such as RMTP [17], RMTP II [22], etc. However, these solutions either do not scale well, or are inflexible (e.g., they employ static hierarchies). Recently, a new class of solutions with the assistance of the network elements (routers) has emerged. Router support in network can often help to improve the performance of reliable multicast protocols.

Major proposed schemes using router support can be roughly divided into two categories.

1) Router Support to richer multicast forwarding semantics

This category of schemes uses minimal router support to direct retransmission request (e.g., NAK) to proper repliers, thus, reducing feedback implosion. These protocols include LMS [37], search party [38], and RMCM [39]. The operations of these protocols can be summarily described as: when a loss occurs on a link of the multicast distribution tree, all receivers downstream to the link suffer losses and require retransmissions. The router will redirect retransmission request to a replier link immediately upstream of the loss. After receiving the retransmission request, the replier would retransmit the lost packet by multicasting it to the sub-tree downstream from the link where the loss occurs. Multiple NAKs for the same lost packet can be aggregated and restricted by the replier in the sub-tree suffering the loss.

2) Using Active Routers for Reliable Multicast

This category of schemes uses active routers (or active servers collocated with routers) for NAK suppression, partial multicast, and local recovery. In these schemes, active routers can construct an efficient logical tree for feedback control and retransmission automatically by using the underlying multicast routing tree directly. Thus, routers actively take part in the reliable multicast protocols. The tradeoff is the extra burden on the network, because active routers must maintain soft state of retransmission or buffer/cache packets. These protocols include ARM [40], AER [41], PGM [42], OTERS [43], and RMANP [44].

In other several subsections, we will describe LMS, PGM and ARM in detail, through which we will better understand this class of protocols with routers assistance.

3.4.2 LMS (Lightweight multicast service)

Light-weight Multicast Services (LMS) is a router-assisted scheme created to address the problem of scalable reliable multicast. The scheme defines a set of forwarding services at the routers that help steer control messages to the surrogates (i.e. repliers which is some selected receivers), and targets at the replier to do NAK aggregation and retransmission. Thus, most of the problems associated with reliable multicast mentioned in section 3.2 could be efficiently solved with minimal assistance from the routers. In addition, since the forwarding is done at the network layer, these services do not need to peek into higher layers and thus avoid layer violation.

Routers enhanced with LMS, in addition to their regular duties perform the following additional tasks:

1) Replier selection

Potential repliers (receivers) advertise their willingness to serve as repliers for a particular *(Source, Group)* pair with their local router. Routers propagate these advertisements upstream. Before propagating the message upstream, a router selects one of its downstream interfaces (based on an application-defined metric) as the replier interface. When all routers have received advertisements, the replier state is established. Replier state is soft state which provides robustness and guards against replier and link failures.

2) NAK forwarding

LMS routers forward NAKs hop-by-hop according to the following rules: a NAK from the replier interface is forwarded upstream; a NAK from a non-replier interface (including the upstream interface) is forwarded to the replier interface. However, a NAK from a non-replier down multicast stream interface marks this router as the "turning point" of that NAK. Note that by definition, there can be only one turning point for each NAK but the same turning point may be shared by multiple NAKs. Before forwarding a NACK, the turning point router inserts in the packet the addresses of the incoming and outgoing interfaces, which we call the "turning point information" of the NAK. This information is carried by the NAK to the replier. An example about the process of the request is shown in Figure 3.2.

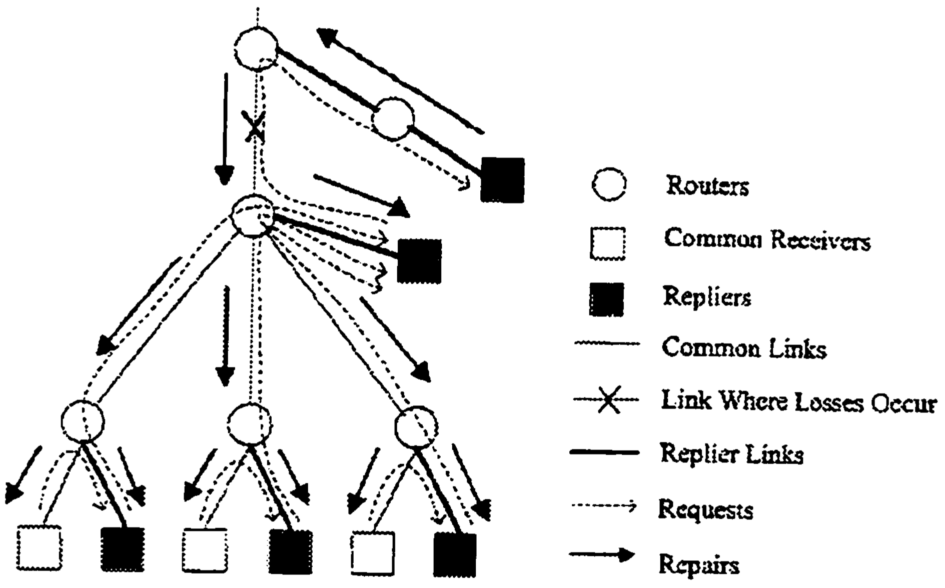


Figure 3. 2 Request and Repair Procedure in LMS

3) Directed multicast (DMCAST)

Repliers use DMCAST to perform fine-grain multicast. A replier creates a multicast packet containing the requested data and addresses it to the group. The multicast packet is encapsulated into a unicast packet and sent to the turning point router (whose address was part of the turning point information) along with the address of the interface the NACK originally arrived at the turning point router. When the turning point router receives the packet, it decapsulates and multicasts it on the specified interface. An enhanced version of DMCAST may allow repliers to specify more than one interface that the packet should be directed to send on. An example about the process of retransmission is shown in Figure 3.2.

Recently, LMS has added incremental deployment methods to their specification in order to be deployed in an incremental fashion on the Internet, due to the scale and inherent heterogeneity of the Internet. These methods help LMS more efficiently improving the performance of reliable multicast.

3.4.3 PGM (Pragmatic general multicast)

PGM (Pragmatic General Multicast) [42] is a recent proposal made by Cisco Systems and Microsoft. It is currently published as a working document of the IETF (Internet draft). The protocol employs "network elements" (routers) to fuse NAKs and constrain retransmissions to save bandwidth, which means that routers do some process of transport level.

PGM [42] has a few data packets, ODATA (original content data), NCF (NAK confirmation), RDATA (retransmission (repair)), SPM (source path message), and NAK (selective negative acknowledgment). The first four packets of them flow downstream in the multicast tree, and NAK packets flow upstream toward the source.

We describe PGM from the following several aspects:

1) Source Path State Establishment

The source will periodically multicast out a Source Path Message (SPM) along the multicast tree to establish source path state for a given source and session in the network. When forwarding a SPM to its downstream nodes, a PGM router will include its own address into the SPM. In this way, a PGM router or receiver can know the address of its upstream PGM router.

PGM routers use this state information to determine the unicast path back to the source for forwarding NAKs. SPMs also alert receivers that the oldest data in the transmit window is about to be retired from the window and will thus no longer be available for repairs from the source.

2) NAK Generation

Upon detecting a packet loss, a receiver will set a back-off timer. When the timer expires, the receiver will generate a NAK for the lost packet and unicast it to its nearest upstream PGM router.

3) NAK Aggregation and Suppression

NAKs are unicast from PGM router to PGM router. When receiving a NAK, a PGM router will add the interface from which the NAK arrives to the repair interface list for the lost packet. . Each PGM router keeps forwarding NAKs until it sees an NCF or RDATA, which indicates that a repair is being sent. This reason that an NCF is sent by multicaste (even limited to the interface on which the corresponding NAK was received) instead of being sent by unicast, is to prevent redundant NAKs. This point will be explained below.

For NAK suppression, the PGM router will immediately multicast a NAK confirmation (NCF) packet along that interface. When a receiver receives the NCF before its timer expires, it will cancel the timer, and no NAK will be generated. When a downstream PGM router receives the NCF, it will stop the propagation of the NCF, and meanwhile it will refrain from forwarding any new NAK for the lost packet to the upstream PGM router.

For NAK Aggregation, the PGM router will not forward a NAK upward if it has forwarded a NAK for the same lost packet upward before.

4) Retransmission of the Data Packet

When the sender receives a NAK, it will first multicast a NCF out, just like what a PGM router does. Then it will multicast the repair packet along the interface from which the NAK arrives. When a PGM router receives the repair packet, it will multicast the packet along all interfaces in the repair interface list for the packet, which will eliminate the transmission of repair data to parts of the distribution tree where the repair is not needed. A router that is not PGM capable will simply forward the multicast packet (including the NCF and the Repair Packet) on all downstream interfaces.

3.4.4 ARM (Active reliable multicast)

ARM [40] is a NAK-based scheme that utilizes active routers at strategic locations to protect the sender and network bandwidth from unnecessary feedback and repair traffic. ARM provides the active services in the network similarly to PGM, except caching data for possible retransmission.

ARM assumes that the forward multicast paths correspond to reverse unicast paths. It caches information in routers' soft state to aggregate NAKs and limit the delivery of repairs. We introduce the scheme from the three aspects:

1) Caching Data at Routers

Active routers (selectable) perform best-effort caching of multicast data packets for possible retransmission. After detecting a loss, a receiver sends a NAK to the source.

When the NAK reaches an active router on the path, the router will retransmit the requested packet if that packet is in its cache; otherwise, it will consider forwarding the NACK towards the sender. The holding time of a data packet in cache depends on the sending rate of data and the max RTT between the sender and the "farthest" receiver downstream.

In fact, caching deals with a tradeoff between network-based storage and bandwidth. Data caching function significantly reduces the recovery latency for distance receivers and lossy links. It also protects the sender and bottleneck links from retransmission requests and repair traffic.

2) Processing of NAK packets

Each active ARM router maintains a NAK record and a REPAIR record for each loss that it is handling for a short amount of time in cache. The NAK record is used to suppress subsequent duplicate NAKs of the same packet, and the REPAIR record is mainly used to suppress NAKs sent by receivers before they receive a repair that is in transit. In general, the holding time of a NAK record in cache depends on how soon the router expects to receive the corresponding repair, and the holding time of a REPAIR record should be approximately one RTT from the router to the farthest receiver downstream.

When the NAK packet reaches an active ARM router, the router first checks REPAIR record of this NAK. If the REPAIR record indicates that the requested repair has just been forwarded down the link on which the NAK arrived, the router will drop this NAK. If the requested repair has not been forwarded down the corresponding link, and the requested repair is in the router's cache, the router retransmits the repair and modifies the REPAIR record. Otherwise, the route subscribes the originator of the NAK to a subsequent transmission of a repair and modifies the NAK report, which is the necessary preparation for scoped retransmissions; it forwards only the first NAK for this repair to the sender, according to the NAK report of the lost data packet.

3) Processing of Retransmission Packets

When the NAKs of a multicast data packet reach the router, a subscription bitmap will be created and cached to determine outgoing link to forward subsequent repairs. When an active router obtains a repair packet from the upsteam active router or in the cache of this router, the active router will first look up the corresponding subscription bitmap in the NAK report of this lost packet, then partial multicast the repair packet to these outgoing links which the subscription bitmap indicates. This guarantees that the repair packet is scoped the portion of the multicast group experiencing loss.

3.5 Summary

In this chapter, we gave a definition of reliability for multicast and studied mechanisms, including FEC and ARQ, to provide reliability. In addition, we also studied some router-assisted mechanisms for reliable multicast in detail. In general, these mechanisms make design tradeoff between bandwidth and latency.

In this thesis, I will focus on the buffering and caching techniques of router for reliable multicast, and the data caching of router in my simulation is similar to the aforementioned active service in ARM [40]. Chapter 4 will present this simulation in detail.

Chapter 4

Router Buffering and Caching Techniques for Multi-Session Reliable Multicast

This chapter first presents the details of the simulation concerning buffering, forwarding and caching at the routers for the proposed multi-session reliable multicast. We perform the performance comparisons of three different forwarding buffer allocation policies using ARQ for loss recovery. Also we study the effect of network elements (routers) with caching and no caching on the performance of multi-session reliable multicast.

These three allocation policies of forwarding buffer and caching buffer are No-Split, Uniform-Split and Flexible-Split, which will be described hereinafter.

4.1 Terminology

The following terms have special significance for this simulation:

- *Buffer*

A temporary data storage area that compensates for a difference in data transfer rates and/or data processing rates between sender and receiver.

- ***Buffering***

Buffering is a common technique to improve efficiency of the system in data transfer which involve several I/O devices with different speeds and different transfer data sizes (bytes, blocks).

- ***Cache***

Generally a small block of fast memory that sits between either 1) a smaller, faster chunk of memory and a bigger, slower chunk of memory, or 2) a processor and a bigger, slower block of memory. This is to provide a bridge from something that's comparatively very fast to something that's comparatively slow.

Cache and caching buffer are thought to be the same thing here. Note that caching buffer is mainly used for data buffering in this simulation, and the cached packets can partly recovery the lost data packets of receiver so as to decrease the number of network retransmission.

- ***Caching***

Caching is an optimization technique that is used whenever a process has to compute the value of a function over and over again and it is faster to remember previous values than to compute them over again.

In this simulation, caching means the packet is still buffered in the caching buffer of this router for a while after it leaves the router.

- ***Cache Hit***

A cache hit occurs when a requested packet is found in the cache in this simulation.

- ***Unicast Routing Table (URT)***

This table specifies the outgoing interface for a data packet to next hop towards the destination.

- ***Multicast Routing Table (MRT)***

This is the multicast topology table, which is typically derived from the unicast routing table. In this scheme, the MRT is used to decide where to send data packets. A secondary function of the MRT is to provide routing metrics for destination addresses.

- ***Upstream***

Towards the root of the tree. The root of tree is the source.

- ***Downstream***

Away from the root of the tree.

- ***Iif***

Incoming interface of certain node.

- ***Oif***

Outgoing interface of certain node.

4.2 Introduction to Router Architecture

Router is a network device that can handle multiple protocols by sending data between dissimilar networks. A basic architectural of an IP router is given in Figure 4.1, which includes the control card (which holds the CPU), the router back plane, and interface cards. The CPU in the router typically performs route processing (i.e., path computation, routing table maintenance, and reachability propagation). It runs whichever routing protocols needed in the router. The interface cards consist of adapters that perform inbound and outbound packet forwarding. The router back plane is responsible for transferring packets between the cards.

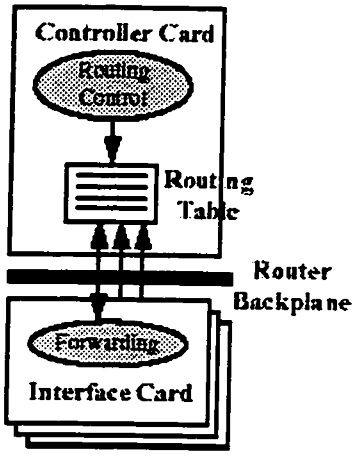


Figure 4.1 Basic Router Architecture

Router must perform two fundamental tasks, routing processing and packet forwarding. The routing process collects information about the network topology and creates a forwarding table. The packet-forwarding process copies a packet from an input interface of the router to the proper output interface based on information contained in the forwarding table (see Figure 4.2). In the process of packet forwarding, firstly the router accepts the arriving packet on an incoming link, and lookups packet destination address in the forwarding table, to identify outgoing port(s). Next the router manipulate packet header (decrement TTL, update header checksum, etc.), sends packet to the outgoing port(s), classifies and buffers packet in the queue. At last, it transmits the packet onto the outgoing link.

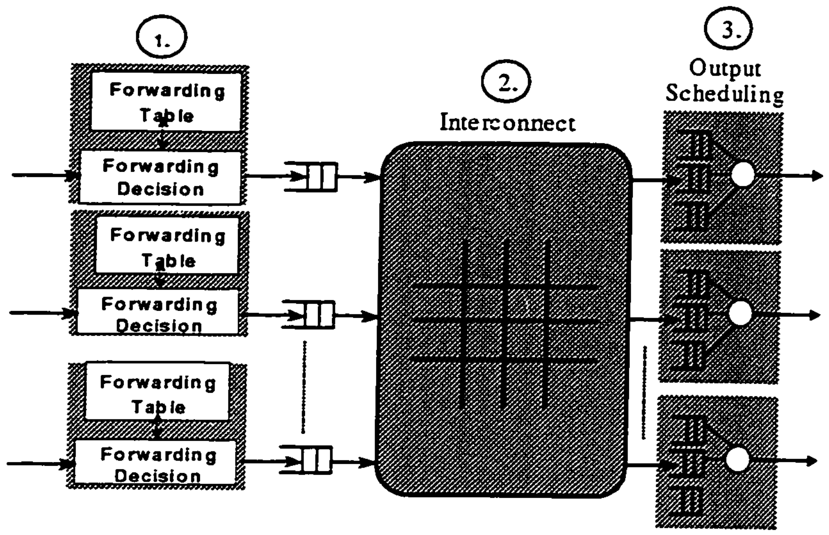


Figure 4.2 Packet Forwarding Processing

In the part of output scheduling, usually there are two basic queuing techniques: input queuing and output queuing (see Figure 4.3). Input queuing means that packets are queued in input buffer, while output queuing means that packets are queued in output buffer. In this simulation, output queuing is adopted.

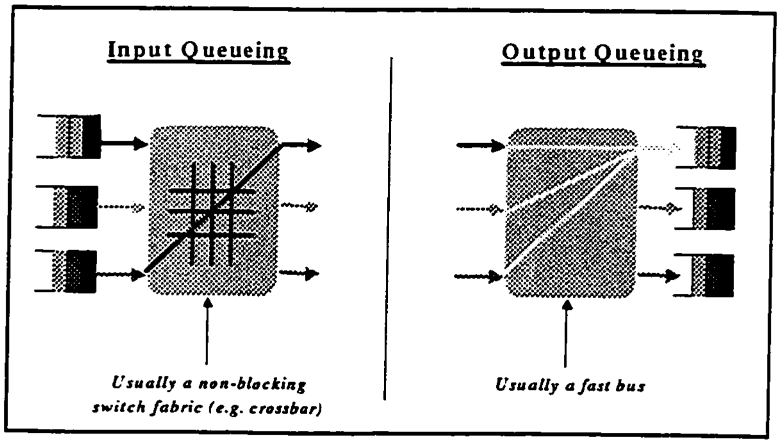


Figure 4.3 Two Basic Queuing Techniques

4.3 Simulation Model

4.3.1 Network Model and Topology

4.3.1.1 Network Model

We adopt a typical network model as shown in Figure 4.4. The network consists of multiple multicast trees/sessions, where each tree has a sender at its root and several receivers at its leaves. Receivers are connected to high-bandwidth backbone network through edge routers.

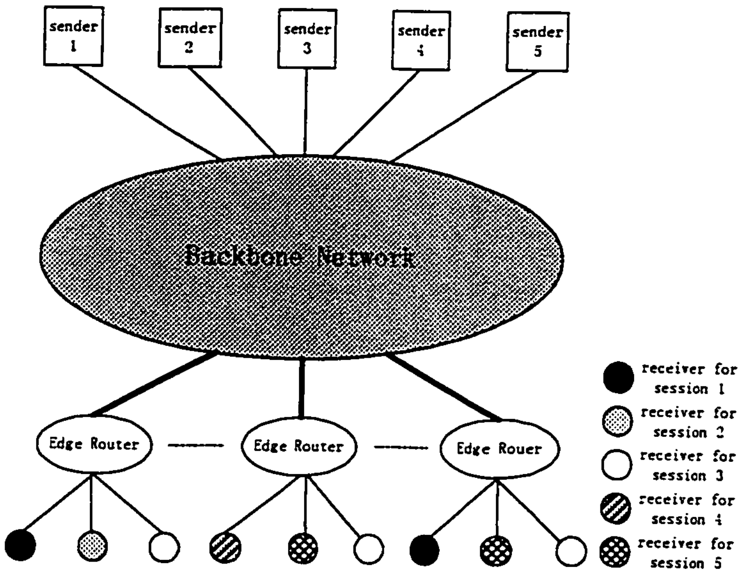


Figure 4.4 Network Model

We assume that the network provides a "best-effort" service model, and that the underlying network is unreliable and packets can be lost or delayed. The end-points must ultimately be responsible for the reliable transport of data packets. However, they can take advantage of network-based processing and storage to improve end-to-end performance and scalability for certain applications. Some intermediate routers will cache data packets for possible retransmission. This point is similar to the role of active router in ARM [40], but here we are just concerned with the cached data packets, other aspects will not be considered.

4.3.1.2 Topology of Multicast Network and the Specific Source Trees

The topology of the simulation is shown as Figure 4.5, in which there are 11 backbone routers, 11 edge routers, 5 sources, 14 receivers (it is enough to have at least 1 receiver for the same session under the same edge router). We also present the detailed multicast path with multicast tree for each session in Figure 4.6. Tables 4.1 to 4.4 give the corresponding routing tables.

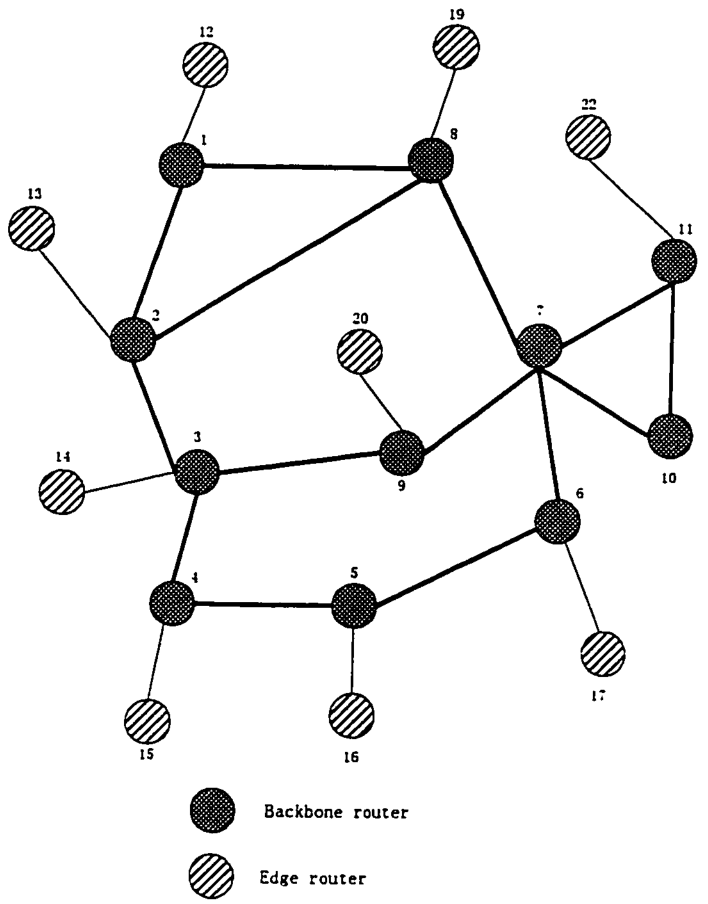


Figure 4.5 Hierarchical Topology of Multicast Network

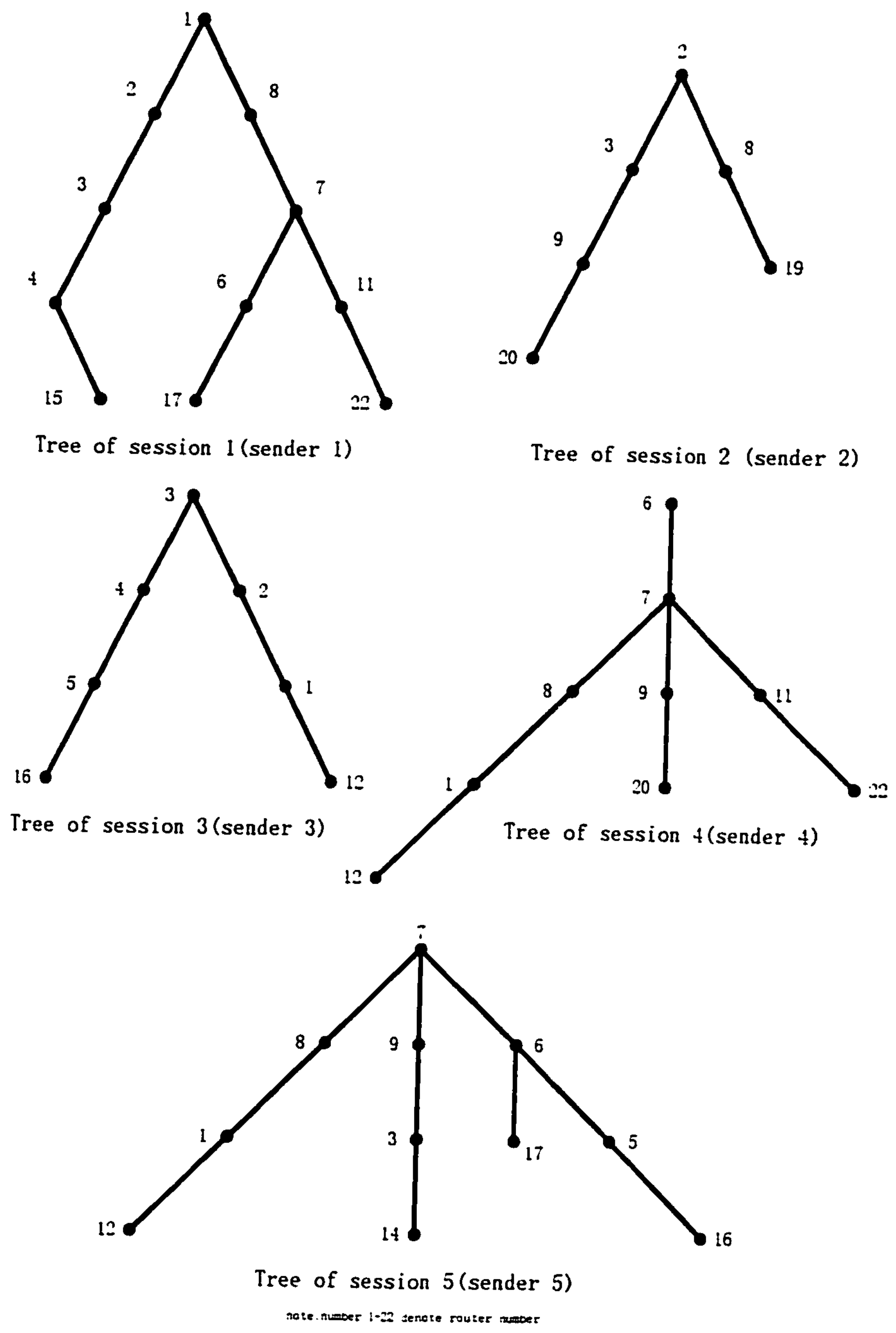

1
2
8
3
7
4
6
11
15
17
22
Tree of session 1(sender 1)
2
3
8
9
19
20
Tree of session 2 (sender 2)
3
4
2
5
1
16
12
Tree of session 3(sender 3)
6
7
8
9
11
1
20
22
12
Tree of session 4(sender 4)
7
8
9
6
1
3
17
5
12
14
16
Tree of session 5(sender 5)
note.number 1-22 denote router number

Figure 4.6 Specific Source Trees for the Simulation

In this simulation, routers use multicast routing table (Tables 4.1) to find outgoing interfaces for multicast data packet of different session. For example, in router 1, the outgoing interface for session 1 is OIF 1 and OIF 2, and accordingly its next hop node is router 2 and router 8; the outgoing interface for session 3, 4, 5 is OIF 3, and their next hop node is router 12; there is no multicast traffic passing through the OIF 4 of router 1.

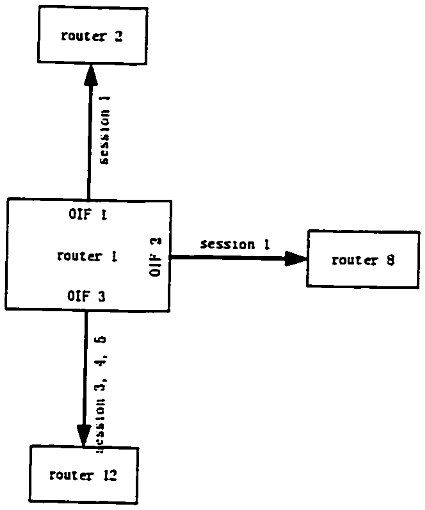


Table 4.1 Multicast Routing Table (Backbone Router)

To / From	OIF 1		OIF 2		OIF 3		OIF 4	
	Next router	Session no.	Next router	Session no.	Next router	Session no.	Next router	Session no.
Router 1	2	1	8	1	12	3, 4, 5	-----	-----
Router 2	3	1, 2	8	2	1	3	-----	-----
Router 3	4	1, 3	9	2	2	3	14	5
Router 4	15	1	5	3	-----	-----	-----	-----
Router 5	16	3, 5	-----	-----	-----	-----	-----	-----
Router 6	17	1, 5	7	4	5	5	-----	-----
Router 7	6	1, 5	11	1, 4	8	4, 5	9	4, 5
Router 8	7	1	19	2	1	4, 5	-----	-----
Router 9	20	2, 4	3	5	-----	-----	-----	-----
Router10	-----	-----	-----	-----	-----	-----	-----	-----
Router11	22	1, 4	-----	-----	-----	-----	-----	-----

OIF : outgoing interface From: from a router(source router) To: to next neighbor router

Table 4.2 indicates the relationship among edge router, its hosts, and the sessions. For example, for edge router 12, in these hosts that locate at its downstream link, receiver 1 (host 1) joins into session 3 (group 3), receiver 2 joins into session 4, and receiver 3 joins into session 5, but no hosts intend to join into session 1 and session 2.

Table 4.3 indicates the corresponding relationship between the source and its neighbor backbone router.

Table 4.2 Multicast Routing Table (Edge Router)

to s from	Session 1	Session 2	Session 3	Session 4	Session 5
Router 12	-----	-----	Receiver 1	Receiver 2	Receiver 3
Router 13	-----	-----	-----	-----	-----
Router 14	-----	-----	-----	-----	Receiver 4
Router 15	Receiver 5	-----	-----	-----	-----
Router 16	-----	-----	Receiver 6	-----	Receiver 7
Router 17	Receiver 8	-----	-----	-----	Receiver 9
Router 18	-----	-----	-----	-----	-----
Router 19	-----	Receiver 10	-----	-----	-----
Router 20	-----	Receiver 11	-----	Receiver 12	-----
Router 21	-----	-----	-----	-----	-----
Router 22	Receiver 13	-----	-----	Receiver 14	-----

s:session no. from: from an edge router to: to the according receiver

Table 4.3 Sender and Its Neighbor Backbone Router

Source 1	Source 2	Source 3	Source 4	Source 5
Router 1	Router 2	Router 3	Router 6	Router 7

Table 4.4 is the reverse unicast routing table used for NAK packets. It indicates the next hop node for the NAK packets of certain session. For example, in router 1, the next node for NAK packets of session 1 is node 31(i.e. source 1); the NAK packets of session 2 should not pass through router 1; the next node for the NAK packets of session 3 is router 2; the next nodes for the NAK packets of session 4 and session 5 are the same router 8.

Table 4.4 Reverse Unicast Routing Table for Each Session (used for NAK packets)

to / s / from	1	2	3	4	5
1	31	------	2	8	8
2	1	32	3	------	------
3	2	2	33	------	9
4	3	------	3	------	------
5	------	------	4	------	6
6	7	------	------	34	7
7	8	------	------	6	35
8	1	2	------	7	7
9	------	3	------	7	7
10	------	------	------	------	------
11	7	------	------	7	------
12	------	------	1	1	1
13	------	------	------	------	------
14	------	------	------	------	3
15	4	------	------	------	------
16	------	------	5	------	5
17	6	------	------	------	6
18	------	------	------	------	------
19	------	8	------	------	------
20	------	9	------	9	------
21	------	------	------	------	------
22	11	------	------	11	------

s: session no. from: from a router to: to upstream router or senders
1-11: backbone routers 12-22: edge routers 31-35: sender no. 1-5

4.3.1.3 Function of Each Part

There are four main parts in our multicast network, which are sender (source), backbone router, edge router, and receiver. For each part, there exist the following functions within the simulation program.

1) Function of Sender

- Generate data packets and forward them to backbone router.
- When receiving NAK, sender retransmits the packet to the downstream router by unicast.

2) Function of Backbone Router

- Multicasting data packets to the downstream routers according to multicast routing table;
- When receiving NAK from the downstream router, either it retransmits directly the requested packet to the receiver (this router is equipped with caching buffer and the packet is cached at that time) or it forwards the NAK packets to the upstream backbone router or sender;
- Handling data buffering and forwarding according to the three different policies in this thesis.

3) Function of Edge Router

- The function of edge router is similar to the backbone router except that no data is cached. And edge router is responsible for forwarding data to or from the receivers of certain local domain.

4) Function of Receiver

- It checks sequence number of packets. If it detects a loss, send NAK to its edge router.

4.3.2 Queuing Discipline

In this simulation, we adopt two queuing disciplines: FIFO and Fair Queuing. The No-Split forwarding buffer policy uses FIFO Queuing, and the other two policies use Fair Queuing. We will introduce the two queuing disciplines and present their advantage and disadvantage.

1) FIFO Queuing

Routers traditionally have used a FIFO queuing discipline. A single queue is maintained at each output port. When a new packet arrives and is routed to an output port, it is placed at the end of the queue. As long as the queue is not empty, the router transmits packets from the queue, taking the oldest remaining packet next.

There are several drawbacks to the FIFO queuing discipline:

(1) No special treatment is given to packets from flows that are of higher priority or are more delay sensitive. If a number of packets from different flows are ready to forward, they are handled strictly in FIFO order.

(2) If a number of smaller packets are queued behind a long packet, then FIFO queuing results in a larger average delay per packet than if the shorter packets were transmitted before the longer packet. In general, flows of larger packets get better service.

(3) A greedy TCP connection can crowd out more other TCP connections. If congestion occurs and one TCP connection fails to back off, other connections along the same path segment must back off more than they would otherwise have to do.

2) Fair Queuing (FQ)

A router maintains multiple queues at each output port. In general maintaining one queue for each source. With fair queuing, each incoming packet is placed in the appropriate queue. The queues are serviced in round-robin fashion, taking one packet from each non-empty queue in turn. Empty queues are skipped over.

This scheme is fair in that each busy flow gets to send exactly one packet per cycle. Further, this is a form of load balancing among the various flows. Also note that there is no advantage in being greedy. A greedy flow finds that its queues become long, increasing its delays, whereas other flows are unaffected by this behavior.

4.3.3 Buffer Management Model

In this simulation, one of the tasks aims at studying of buffer (the buffer of router output port) sharing for multi-sessions reliable multicast. Several flows of packets may share a common pool of buffers of an output port. Buffer management sets the buffer sharing policy and decides which packet should be discarded when the buffer overflows.

4.3.3.1 Buffer Allocation Algorithms

So-called buffer allocation algorithm means the buffer sharing policy of output port. Any given node has a number of I/O ports to other nodes or end systems. There are two buffers at each port: input and output buffer. We assume that all routers are output-buffered routers. There is a fixed-size output buffer associated with each output port, and the buffer size is the same for all output ports of all routers.

The forwarding buffer allocation policies focus on the problem how the buffer of each port is partitioned for different session whose stream is departing from the port. In my scheme, three policies are adopted: No-Split policy, Uniform-Split policy and Flexible-Split policy.

No-Split forwarding buffer allocation policy means the buffer is not partitioned and all sessions that go through the port to next node share the buffer of this port. The packets are stored in the buffer with their reaching orders (first in first stored), regardless of the sessions to which the packets belong. The maximum length of this queue is equal to the maximum size of this output buffer.

In contrast to No-Split policy, other two policies mean that the buffer of this port is split into several small buffers according to different sessions that go through the port. In other words, each output port of any node maintains several different queues for different sessions. For Uniform-Split policy, each queue in certain output buffer has equal maximum length by uniformly splitting the output buffer according to the different sources whose stream is departing from the port. For Flexible-Split policy, the maximum length of each queue is unequal by flexibly splitting the output buffer according to the different sources and is proportional to the input traffic load of this source. The formula of each policy is listed below.

- No-Split

$$Max\_Ql = Max\_Bs \tag{4.1}$$

Where Max_Ql is the maximum length of the queue, and Max_Bs is the maximum forwarding buffer size of an output port in router.

- Uniform-Split

$$Max\_Ql = \frac{Max\_Bs}{Num\_Se} \tag{4.2}$$

Where *Max_Ql* is the maximum length of each queue in an output port, *Max_Bs* is the maximum forwarding buffer size of an output port in router, and the *Num_Se* is the number of sessions through the output port.

- Flexible-Split

$$Max\_Ql_i = Max\_Bs * \frac{TL_i}{\sum_{j=1}^{N} TL_j} \tag{4.3}$$

Where Max_Ql_i is the maximum queue length of session *i* in an output port, *Max_Bs* is the maximum forwarding buffer size of an output port in router, TL_i is input traffic load of session *i* advertised into multicast packet, and N is the number of sessions through the output port.

4.3.3.2 Discarding (Drop) Policy

Drop policy decides which packet should be discarded when a certain buffer overflows.

The Internet Protocol (IP) architecture is based on a connectionless end-to-end packet service using the IP. The advantages of its connectionless design, flexibility, and robustness, have been amply demonstrated. However, these advantages are not without cost: careful design is required to provide good service under heavy load. In fact, lack of attention to the dynamics of packet forwarding can result in severe service degradation.

There are many policies of buffer management in the Internet, for example: tail drop, random drop on full, drop front on full, random early detection (RED), differential dropping (RIO), etc. In this simulation, we adopt the policy of tail drop in order to simplify the simulation.

Tail Drop, the traditional technique for managing router queue lengths, sets a maximum length (in packets) for each queue, accepts packets for the queue until the maximum length is reached (i.e., the queue is full), then drops subsequent incoming packets until the queue decreases because a packet from the queue has been transmitted.

4.3.4 Data Caching Model

Since edge routers connect the local area network (LAN) to the wide area network (WAN), more handling capacity is paid for routing LAN as well as WAN packets. Thus, the handling speed of packets is lower than backbone router if CPU speed is the same. Moreover, the backbone network is bandwidth-rich, but the access links to backbone network have typically slow-speed. The above two points lead to that fact that most packet losses occur on the links from backbone router to edge router, i.e. "edges" of the network [56].

Due to the above two reasons, we just utilize caching techniques in these backbone routers which are connected to edge routers for local recovery. Only these data packets, which are departing from this backbone router to the next edge router, are cached.

The caching buffer size of each router is fixed and limited. We adopt FIFO (first in first out) with a holding time constraint as the caching buffer replacement policy. This is to avoid the problems that when caching buffer is not sufficient, a packet cached at the router is possible to be replaced with another packet before it gets a chance to be sent as a repair downstream if TTL is low, but on the other hand if this TTL is too high, it may result in more overflows. When the packet is being forwarded to the next node, it is stored in the caching buffer with TTL (Time to Live) which is the holding time in caching buffer.

4.3.5 Input Traffic Load and Loss Model: Uniform Random Generator

Source input model is Bernouli model and follows the uniform distribution between 0 and the traffic load of the source. We call a uniform distribution function to generate a uniform random number at each iteration. If the random number is less than the given traffic load of this source, the function return the value 1 and it means this source can generate a packet at that iteration; otherwise no packet is generated. For example: the traffic load of source 1 =0.8 and the generated random number at certain iteration=0.68, then the source 1 can generate a packet at this iteration.

Buffer overflow and bad link state can generate packet loss. In this simulation, both the losses will be considered. The packet loss due to buffer overflow is dependent, and the link loss is assumed to be independent. The loss occurrence on links follows the uniform distribution between 0 and the loss rate of the link. The process is similar to the input process. Before a packet passes a link, we call a uniform distribution function to generate a uniform random number. If the number is less than the loss rate of the link, loss happens on this link and this packet is discarded. Otherwise this packet won't be lost on this link at this iteration.

The uniform distribution gives an equal probability for the value of a random variable over a given interval. The continuous uniform distribution function between 0 and a (traffic load of a source or the loss rate of a link) is given by

$$F(x) = \begin{cases} 0 & \text{if } x < 0 \\ \frac{x}{a} & \text{if } 0 \le x \le a \\ 1 & \text{if } a < x \end{cases} \tag{4.4}$$

The corresponding density function is given by

$$f(x) = \begin{cases} \frac{1}{a} & \text{if } 0 \le x \le a \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

4.3.6 Assumptions for this Simulation

In this simulation, we assume the following points:

- All data packets have the same size. The size of forwarding buffer and caching buffer are measured in terms of the number of data packets it can store.
- NAK does not occupy much of the buffer of the routers, because NAK packet is very small compared with a data packet, so it is negligible.
- Packet losses on links are independent and follow uniform distribution. In my simulation, the packet loss rate on link is point to point (each link).
- Each sender at most sends a packet per iteration. Each backbone router at most sends 4 packets at certain link per iteration. Each edge router at most sends 2 packets at certain link per iteration.

4.4 Performance Measurement

Parameters given in this section are used for evaluating the performance of multi-session reliable multicast. Each performance parameter is the averaged value over some input parameters, such as the number of users.

- **Average Forwarding Buffer Overflow**

In routers, each queue length has its maximum limit due to the finite buffer size of each output port. If the rate at which packets arrive and queue up exceeds the rate at which packets can be transmitted, the queue size grows to its limit and extra packets will be lost. That means that buffer overflow happens.

In this simulation, we set a counter to record the number of buffer overflow throughout the whole simulation. The average of the forwarding buffer overflow is calculated by dividing the sum of these overflows by the total iterations and by the sum of the number of output ports of all routers. We give the following formula for average forwarding buffer overflow (FBO).

$$FBO = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{P(j)} O(i,j,k)}{N * \sum_{j=1}^{M} P(j)} \qquad (4.6)$$

Where $O(i, j, k)$ is the number of forwarding buffer overflow at output port k of router j at iteration i, $P(j)$ is the total number of output ports of router j, N is the number of iteration, and M is the number of routers.

- **Average and Variance of Queuing Delay (Length)**

Queuing Delay is defined as the time that a packet needs to wait from entering the queue to departing the queue. The mean queue length, or the average number of packets waiting to be forwarded at output queue is directly proportional to queuing delay. Hence, we use queue length in unit of packet for evaluating the queue delay.

In this simulation, average and variance of the queuing delay can be calculated in two cases. The first case (named: *for all sessions*) is calculated according to the trace of each session in the network (specific source multicast tree). Another case (named: *for all receivers*) is calculated according to the trace from the source to each receiver in the network. The formulas for these two methods are given below.

Method 1: ***for all receivers***

$$AQL = \frac{\sum_{i=1}^{Se}\sum_{j=1}^{Re(i)}\left(\frac{\sum_{k=1}^{Ite}\sum_{l=1}^{Ro(i,j)} QL_{ijkl}}{\sum_{k=1}^{Ite} Ro(i,j)}\right)}{\sum_{i=1}^{Se} \mathrm{Re}(i)} \qquad (4.7)$$

$$VQL = \frac{\sum_{i=1}^{Se}\sum_{j=1}^{Re(i)}\left(\frac{\sum_{k=1}^{Ite}\sum_{l=1}^{Ro(i,j)} QL_{ijkl}}{\sum_{k=1}^{Ite} Ro(i,j)} - AQL\right)^2}{\left(\sum_{i=1}^{Se} \mathrm{Re}(i)\right) - 1} \qquad (4.8)$$

Where AQL is the average queue length, VQL is the variance of the queue length, Se is the number of sessions, $Re(i)$ is the number of receivers for session i, Ite is the number of iterations, $Ro(i, j)$ is the number of routers passed by the packets from the source i to receiver j, and QL_{ijkl} is the queue length of session i in router l along the trace of from source i to receiver j at iteration k.

Method 2: ***for all sessions***

$$AQL = \frac{\sum_{i=1}^{N}\left(\frac{\sum_{l=1}^{Ite}\sum_{j=1}^{Ro(i)}\sum_{k=1}^{Po(i,j)} QL_{iljk}}{Ite * \sum_{j=1}^{Ro(i)} Po(i,j)}\right)}{N} \tag{4.9}$$

$$VQL = \frac{\sum_{i=1}^{N}\left[\left(\frac{\sum_{l=1}^{Ite}\sum_{j=1}^{Ro(i)}\sum_{k=1}^{Po(i,j)} QL_{iljk}}{Ite * \sum_{j=1}^{Ro(i)} Po(i,j)}\right) - AQL\right]^2}{N-1} \tag{4.10}$$

Where *AQL* is the average queue length, *VQL* is the variance of the queue length, *N* is the number of sessions, *Ro(i)* is the number of routers along the multicast tree of session *i*, *Ite* is the number of iterations, *Po(i, j)* is the number of output ports of routers *j* for session *i*, and QL_{iljk} is the queue length of session *i* in output port k of router *j* along the multicast tree of session *i* at iteration *l*.

- **Average Caching Buffer Overflow**

There is only one size-limited caching buffer in one router with cache. When caching buffer is full and the cached time of all cached data packets don't exceed the holding time in caching buffer, new packets won't be cached. We call it as caching buffer overflow. This parameter indicates the probability of how many packets can't be cached.

$$CBO = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} BO_{ij}}{N * M} \tag{4.11}$$

Where CBO means average caching buffer overflow, BO_{ij} is the number of caching buffer overflow of router j at iteration i, N is the number of iterations, and M is the number of routers.

- **Cache Hit Probability**

The cache hit probability is the probability that a requested packet will be found in the caching buffer for all routers with caching. We assume only one cache per router. The more the cache hit probability, the more the requested packets will be found, which can save more bandwidth of network and improve the network delay due to decreasing of retransmissions.

$$CBP = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} FO_{ij}}{\sum_{i=1}^{N}\sum_{j=1}^{M} FL_{ij}} \tag{4.12}$$

Where CBP is cache hit probability, N is the number of iterations, M is the number of routers, FO_{ij} is the number of found packets in router j at iteration i, and FL_{ij} is the number of packets which are looked for in router j at iteration i.

- **Average and Variance of End to End Delay**

In this simulation, end to end delay means the total time that a packet takes from its generation to its successful delivery to the final destination. It includes the queuing delay and the propagation delay. The average and variance of end to end delay can be calculated in two methods, as such described in queue delay. Its time units are iterations (or packets).

Method 1: ***for all receivers***

$$AED = \frac{\sum_{i=1}^{N}\left[\frac{\sum_{j=1}^{Pa(i)}\left(TE_{ij} - TB_{ij}\right)}{Pa(i)}\right]}{N} \tag{4.13}$$

$$VED = \frac{\sum_{i=1}^{N}\left[\frac{\sum_{j=1}^{Pa(i)}\left(TE_{ij} - TB_{ij}\right)}{Pa(i)} - AED\right]^2}{N-1} \tag{4.14}$$

Where AED is the average end to end delay, VED is the variance of end to end delay, N is the total number of receivers of all sessions, $Pa(i)$ is the number of packets received by receiver i, TE_{ij} is the end iteration of packet j of receiver i, and TB_{ij} is the begin iteration of packet j of receiver i.

Method 2: ***for all sessions***

$$AED = \frac{\sum_{i=1}^{N}\left[\frac{\sum_{j=1}^{Re(i)}\sum_{k=1}^{Pa(j)}\left(TE_{ijk} - TB_{ijk}\right)}{\sum_{j=1}^{Re(i)} Pa(j)}\right]}{N} \tag{4.15}$$

$$VED = \frac{\sum_{i=1}^{N}\left[\frac{\sum_{j=1}^{Re(i)}\sum_{k=1}^{Pa(j)}\left(TE_{ijk} - TB_{ijk}\right)}{\sum_{j=1}^{Re(i)} Pa(j)} - AED\right]^2}{N-1} \tag{4.16}$$

Where AED is the average end to end delay, VED is the variance of end to end delay, N is the total number of sessions, $Re(i)$ is the number of receivers of session i, $Pa(j)$ is the number of packets of receiver *j*, TE_{ijk} is the end iteration of packet *k* of receiver *j of session i*, and TB_{ijk} is the begin iteration of packet *k* of receiver *j* of session *i*.

- **Average and Variance of NAKs**

NAKs are defined as the total number of NAK packets, which are received at one sender. The parameter is one of important performance criteria in reliable multicast communication, because excess NAKs could lead to the well-known NAK implosion problem which will seriously deprave the performance of reliable multicast.

$$AN = \frac{\sum_{i=1}^{N} Na(i)}{N} \tag{4.17}$$

$$VN = \frac{\sum_{i=1}^{N} \left(Na(i) - AN\right)^2}{N-1} \tag{4.18}$$

Where *AN* is the average number of NAKs received by each source, *VN* is the variance of NAKs, *N* is the number of sources, *Na(i)* is the total number of NAKs which is received by source *i*.

- **Average and Variance of Session Transmission Time**

It is the time during which total packets of certain session are received by certain receiver or by all receivers of certain session. It is a parameter measuring end to end performance, and its time unit is iterations. Two measurement methods are given below.

Method 1: ***for all receivers***

$$ATT = \frac{\sum_{i=1}^{N}(TL_i - TF_i)}{N} \tag{4.19}$$

$$VTT = \frac{\sum_{i=1}^{N}[(TL_i - TF_i) - ATT]^2}{N-1} \tag{4.20}$$

Where ATT is the average session transmission time for each receiver of each session, VTT is the variance of session transmission time, N is the number of receivers of all sessions, TL_i is the end iteration of the latest reaching packet of receiver i, and TF_i is the begin iteration of the first generated packet of receiver i.

Method 2: for all sessions

$$ATT = \frac{\sum_{i=1}^{N}(TL_i - TF_i)}{N} \tag{4.21}$$

$$VTT = \frac{\sum_{i=1}^{N}[(TL_i - TF_i) - ATT]^2}{N-1} \tag{4.22}$$

Where ATT is the average session transmission time for each session, VTT is the variance of session transmission time, N is the number of sessions, TL_i is the end iteration of the latest reaching packet of session i, and TF_i is the begin iteration of the first generated packet of session i.

4.5 Simulation Descriptions

4.5.1 Input Parameters

In this simulation, some parameters of network structure are fixed. Moreover, there are some parameters which can be varied and used to evaluate the performance under different conditions.

Constant Parameters:

- *Sessions:* the number of sessions =5
- *Nodes:* the number of backbone routers =11, the number of edge router=9
- *Sources:* the number of sources=5
- *Receivers*: the total number of receivers =14
- *Maximum Iteration:* the total running time of this simulation =7000 iterations

Variable Parameters:

- *Input Traffic load*

  Input traffic load of each session indicates the number and the frequency of data packets which will be generated. In this simulation, the total number of packets at certain source = 5000*input traffic load of this source. Note that we set Maximum iteration =7000 in order to guarantee that all packets in network can reach their destination at the end of the simulation.

- *Loss Rate*

Loss rate denotes the random loss probability of packet at each link. In this simulation, it can be vary from 0 to 0.02.

- *Forwarding buffer size*

This parameter indicates the maximum buffer size of each output port at router, whose unit is packet. When we simulate the part of three forwarding buffer policies, it is varied from 36 packets to 396 packets. In other part of simulation, we fix it to 30 packets.

- *Caching buffer size*

This parameter indicates the size of caching buffer in one router, whose unit is packet. We vary the value of this parameter from 6 packets to 27 packets in order to study its effect on performance. In other cases, we fix it to 18 packets.

- *TTL of Cached Data*

TTL of cached data means how long the holding time of data packets in caching buffer of router. Its time unit is iteration. When studying its effect on performance, we vary it from 2 iterations to 20 iterations. In other cases, we fix it to 8 iterations.

4.5.2 Data Structure of This Simulation

In the simulation, each data packet contains the head of data packet, which includes fields such as:

- **Data Type:** There are 4 types of data packets, which are Fresh (Type 1), Retransmission (Type 2), NAK (Type 3), and Cache (Type 4). Fresh means the packet that is transmitted successfully to receivers. Retransmission means the packet which is retransmitted by the sender due to the loss of this packet before reaching the receivers. NAK means the packet which is sent by the receivers to inform the sender that which packets are lost and are required to retransmitted. And Cache means the recovery data packet comes from certain router that has cache buffer instead of the sender.

- **Sequence Identifier:** Each packet of each session has a unique identifier.
- **Group (Session) Identifier:** Each session has a unique identifier. In this simulation, there are 5 sessions which are originated by different sources.
- **Begin Iteration:** This field indicates when the data packet is generated.
- **End Iteration:** This field indicates when the data packet reached the destination. The two fields of begin iteration and end iteration can be used to statistic calculation.
- **Input Traffic Load of Session:** This field indicates how much input traffic load this session has. It is useful for flexible allocation of buffer.
- **TTL (Time to Live):** In this simulation, only the cached data packet at routers uses this field to indicate how long the holding time is in the caching buffer.
- **Option:** For NAK type data, this field indicates which receiver sent this NAK packet. For Retransmission and Cache type data, this field indicates to which receiver the packet should be sent. This field is not used by Fresh type data.

4.5.3 Simulation Descriptions

According to the different roles of each part of the network, the simulation includes 4 parts: sender, backbone router, edge router and receiver. The program is written in C Language (Linux operation system). At each iteration, we call these 4 functions: the processing of senders, the processing of backbone routers, the processing of edge routers and the processing of receivers, which is shown in the main flow chart (Figure 4.7).

Besides the above-mentioned parts, we also include the programs which record data and calculate all kinds of performance parameters in order to evaluate the performance of each policy. The details to statistical processing follow the rules of *Performance Measurement in section 4.4*.

In this simulation, we don't consider the establishing procedures of multicast trees, and assume that the multicast trees do not change during the whole multicast session, so that we can focus on the study of different buffering, caching and forwarding policy for the reliable multicast. Based on this assumption, we establish statistic routing table (multicast and unicast), which are shown hereinabove (see Table 4.1 to Table 4.4).

4.5.3.1 Senders Part

Each sender (Figure 4.8) first checks the NAK buffer. If the NAK buffer is not empty, the first stored NAK packet will be handled and the requested packet is retransmitted according to the field of sequence number of this NAK packet, then the counter of NAK packets is decreased by 1 and the sender will wait for the handling of the next iteration. If there is no NAK packets in the NAK buffer, the program calls a uniform distribution random function U(0,1) and generates a random variable X. If X is less than P(ii), the traffic load of sender ii, the sender will generate a new data packet encapsulated with header, and forwards this packet to downstream router. The procedure is shown in Figure 4.8: the flow chart of Sender Processing.

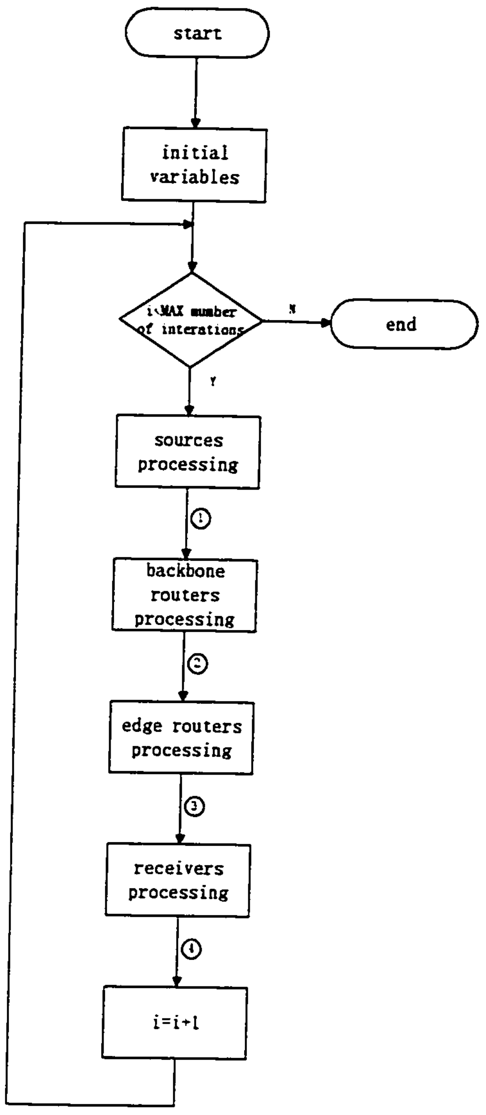


Figure 4.7 Main Program Flow Chart

4.5.3.2 Backbone Routers Part

The simple flow chart of a backbone routers simulation part is shown as Figure 4.9a. Maintaining of queues state is a quite important job for buffer management of router. In this simulation, the queue state includes the following items: the beginning position of the queue, the limit of the queue length, the queue length, and the flag of queue overflow. The last two items of the queue state are dynamically modified as the new packet comes into the queue or the old packet leaves the queue. The simulation depends on the queue state to handle the incoming and departing of the data packets.

Processing description:

1) Check the TTL of all cached packets. If it does not timeout, the TTL will be decreased by 1.

2) Check the NAK buffer (Figure 4.9 b). In case of the NAK buffer being not empty, if the requested packet can be found in cache, we check the unicast routing table to find the output port and load this packet into the buffer of the output port; otherwise, we check reverse unicast routing table and send the NAK packet to its sender. The counter of the NAK packets is decreased by 1. Because priority is given to retransmission packets, the found cached packet is replicated at the head of the queue, and if the queue is full, the cached packet will replace the oldest fresh data packet. The processing of NAK packets is shown as Figure 4.9b.

3) Receive the data packets from every upstream router if necessary (see figure 4.9c). (1) If this packet is fresh, check the multicast routing table to find the output ports according to the session number of the received packet. If the type of this packet is retransmission, check the unicast routing table to find the output port according to the id of receiver requesting this retransmission. (2) If the buffer of the output ports is full, the function of discarding process will be called. In this function, we first check the type of this data packet. If this packet is fresh packet, the new packet is discarded; if this packet is retransmission packet, we discard the oldest packet and store the new packet in the first

position of the queue. (3) If the buffer of the output ports is not full, we store the packet to its exact position in the queue according to the type of this packet. For the fresh packet, it is stored to the tail of the output queue, and for the retransmission packet, it is stored at the head of the output queue.

4) Sending the data packets (see figure 4.9d). The buffer of every output port of a router is handled one by one. (1) Check if the buffer of this output port is empty. If it is empty, this port is ignored and the program goes to handle the buffer of next output port. (2) Otherwise, call the function of packets scheduling to find which queue would be handled. (3) Then check if the loss will happen on its output link. If it is, discard the first packet of this queue, and the size of this queue is decreased by 1. Otherwise, the program goes to next step. (4) Call function of data forwarding and caching. If this packet is either fresh packet or retransmission packet from the source, and there is a cache available for this output port, this packet is forwarded to the downstream router and is cached in the caching buffer at the same time. Otherwise this packet is just forwarded to the next router.

4.5.3.3 Edge Routers Part

The processing of edge router part is similar to the processing of backbone router part except for two different points: there is no function of caching data in edge router; and downstream nodes are receivers. Since we can refer to the part of backbone router, the detailed description of this part will not be given.

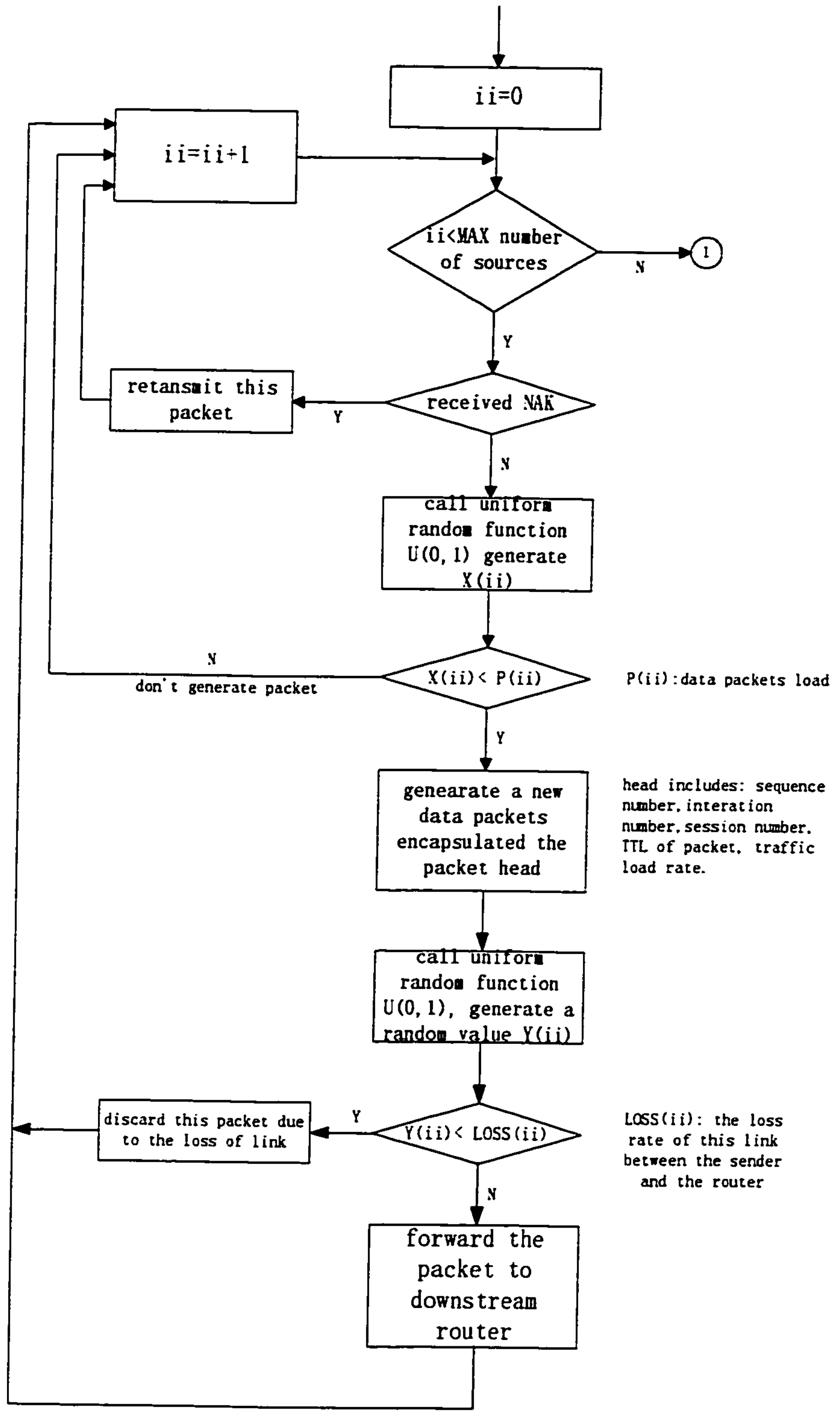
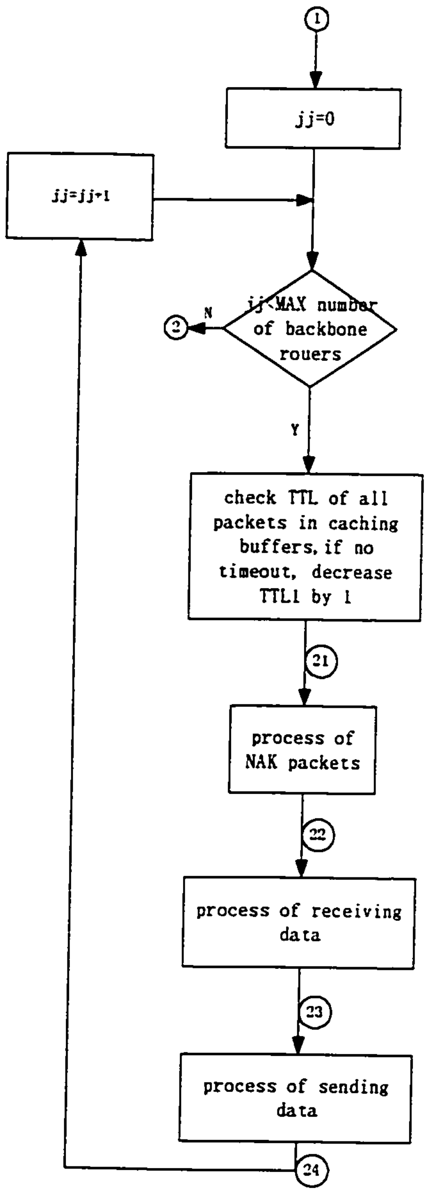

ii=0
ii=ii+1
ii<MAX number of sources
N
1
Y
received NAK
Y
retansmit this packet
N
call uniform random function U(0, 1) generate X(ii)
X(ii)< P(ii)
N
don't generate packet
P(ii):data packets load
Y
genearate a new data packets encapsulated the packet head
head includes: sequence number, interation number, session number, TTL of packet, traffic load rate.
call uniform random function U(0, 1), generate a random value Y(ii)
Y(ii)< LOSS(ii)
Y
discard this packet due to the loss of link
LOSS(ii): the loss rate of this link between the sender and the router
N
forward the packet to downstream router

Figure 4.8 Sender Processing


1
jj=0
jj=jj+1
jj<MAX number of backbone rouers
N
2
Y
check TTL of all packets in caching buffers, if no timeout, decrease TTL1 by 1
21
process of NAK packets
22
process of receiving data
23
process of sending data
24

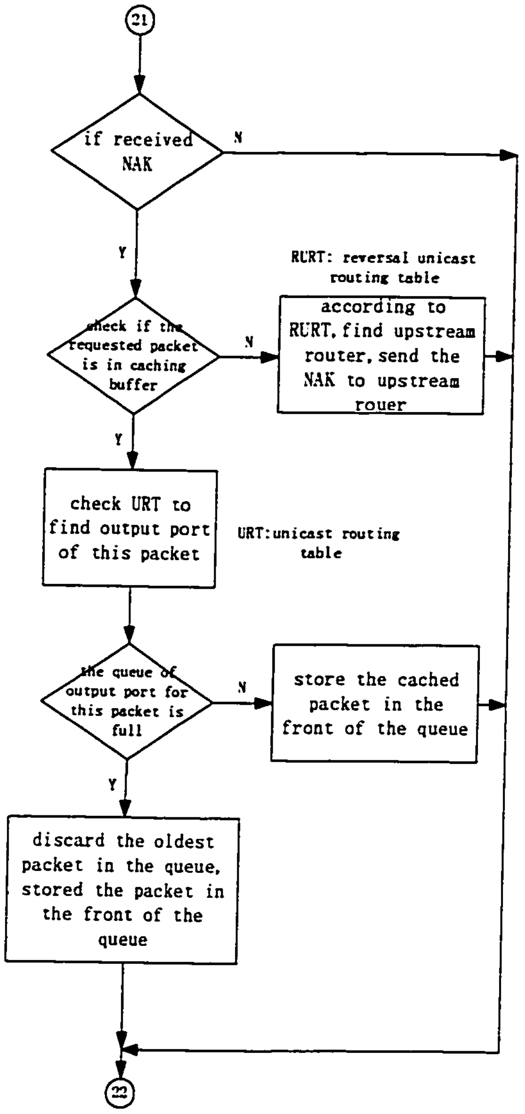

21
if received NAK
N
Y
RURT: reversal unicast routing table
check if the requested packet is in caching buffer
N
according to RURT, find upstream router, send the NAK to upstream rouer
Y
check URT to find output port of this packet
URT:unicast routing table
the queue of output port for this packet is full
N
store the cached packet in the front of the queue
Y
discard the oldest packet in the queue, stored the packet in the front of the queue
22

Figure 4.9a Backbone Router Processing **Figure 4.9b Process of NAK Packets**

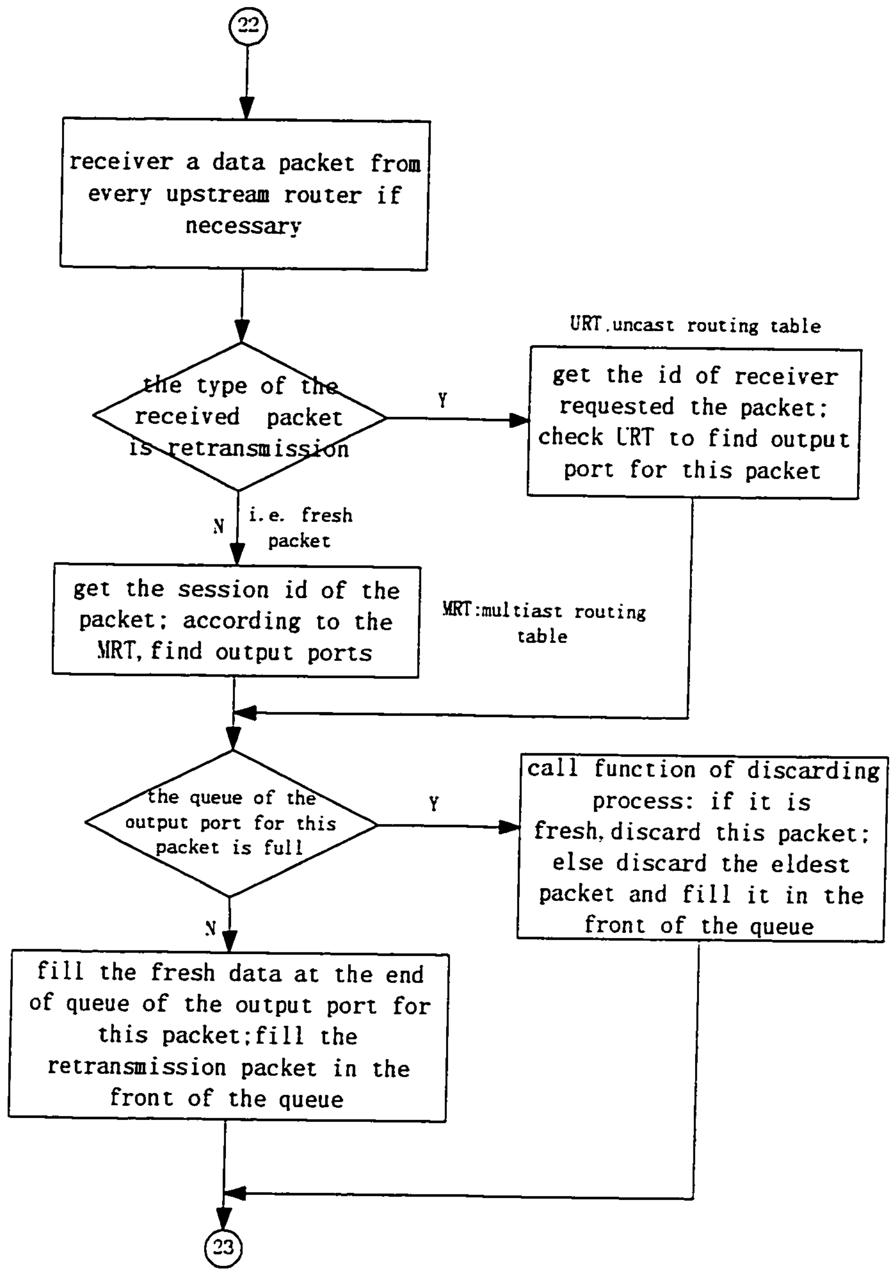

22
receiver a data packet from every upstream router if necessary
URT.uncast routing table
the type of the received packet is retransmission
Y
get the id of receiver requested the packet; check URT to find output port for this packet
N
i.e. fresh packet
get the session id of the packet; according to the MRT, find output ports
MRT:multiast routing table
the queue of the output port for this packet is full
Y
call function of discarding process: if it is fresh, discard this packet; else discard the eldest packet and fill it in the front of the queue
N
fill the fresh data at the end of queue of the output port for this packet;fill the retransmission packet in the front of the queue
23

Figure 4.9c Process of Receiving Data

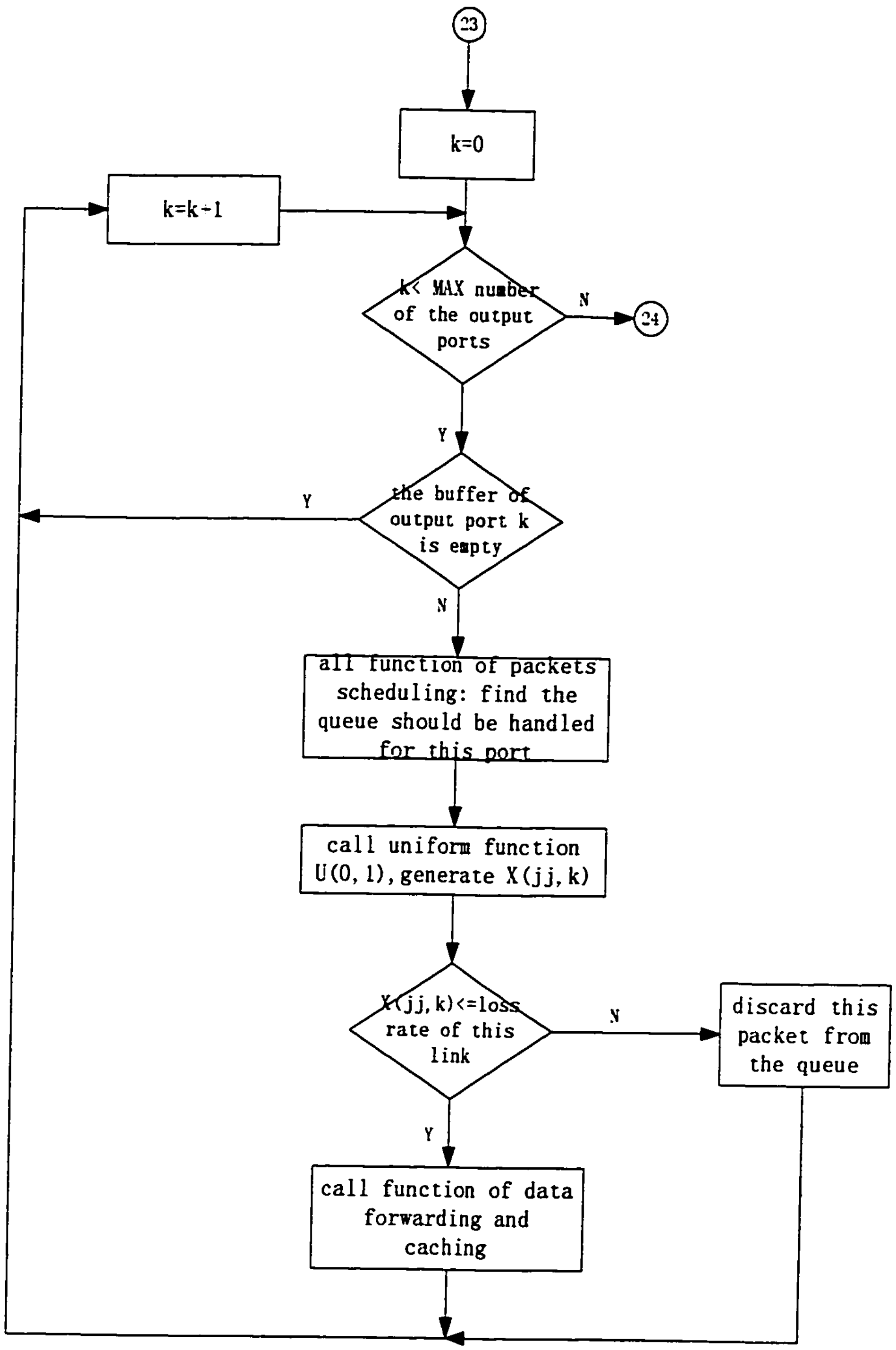

23
k=0
k=k+1
k< MAX number of the output ports
N
24
Y
the buffer of output port k is empty
Y
N
all function of packets scheduling: find the queue should be handled for this port
call uniform function U(0, 1), generate X(jj, k)
X(jj, k)<=loss rate of this link
N
discard this packet from the queue
Y
call function of data forwarding and caching

Figure 4.9d Process of Sending Data

4.5.3.4 Receivers Part

We give the flow chart of receiver part in figure 4.10. Every receiver receives packets from upstream router and stores them into the receiver buffer. Also, the receiver checks the sequence numbers of received packets to see if there are lost packets at every iteration. If the receiver finds any lost packet, it sends NAK for this packet to upstream edge router.

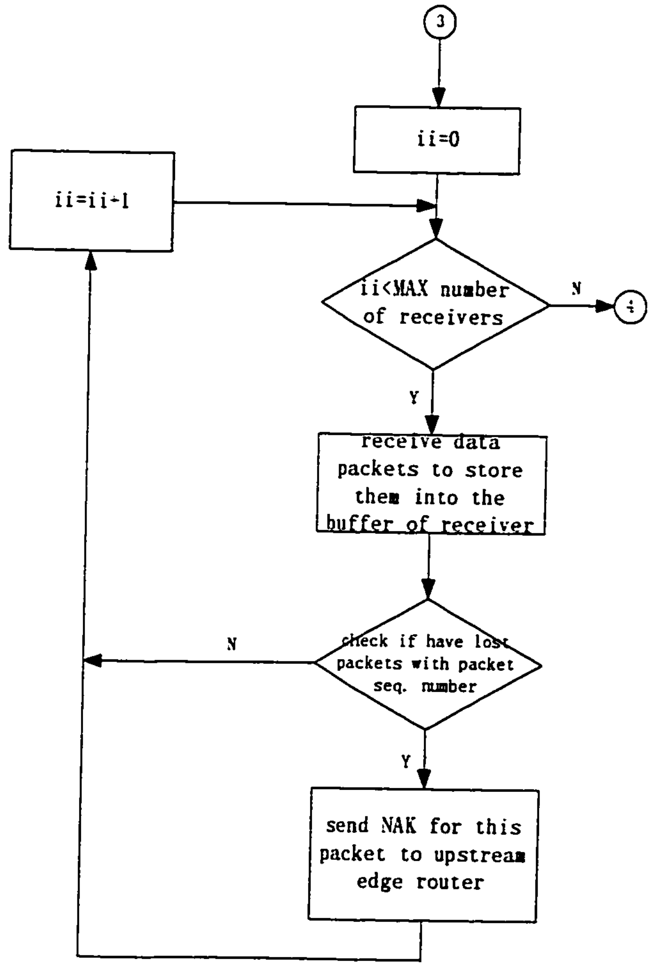

3
ii=0
ii=ii+1
ii<MAX number of receivers
N
4
Y
receive data packets to store them into the buffer of receiver
check if have lost packets with packet seq. number
N
Y
send NAK for this packet to upstream edge router

Figure 4.10 Receiver Processing

4.6 Analysis of Simulation Results

4.6.1 Three Forwarding-Buffer Policies Comparison

In the test for the performance of the three forwarding buffer policies, we won't consider the link loss and caching technique. As shown in Figures 4.9 - 4.14, we vary the forwarding buffer size from 36 to 396 and fix maximum number of iteration=7000, the traffic load of sources 1, 2, 3, 4, 5 are 1, 0.8, 0.8, 0.7 and 0.6 respectively to run the simulation and compare the three policies in terms of end to end delay, average NAKs of source, average forwarding buffer overflow, and queue delay. We give below the detailed comparison of the three polices.

- *End to End Delay*

1) The average end to end delay increases with increasing the forwarding buffer size for the three forwarding buffer polices because the maximum number of packets that can be placed in a queue increases with increasing the buffer size (Figure 4.11-4.12).
2) When forwarding buffer size is same, the end to end delay of No-Split policy is the highest, and that of Uniform-Split policy is the lowest among the three polices.
3) If the forwarding buffer size is sufficient, buffer overflow will not occur; hence the average end to end delay will not change any more with increasing the buffer size. In Figure 4.11, when forwarding buffer size is more than 252, the end to end delay of No-Split policy tends to a constant value with increasing buffer size; when forwarding buffer size is more than 324, Flexible-Split policy tends to a constant value; when forwarding buffer size is more than 396, the end to end delay of Uniform-Split policy tends to a constant value.

4) When forwarding buffer size is big enough for every policy (no buffer overflow), the end to end delay of Flexible-Split policy is equal to that of Uniform-Split policy. But they are lower than No-Split policy.

- ***Forwarding Buffer Overflow***

1) The average forwarding buffer overflow decreases with increasing the forwarding buffer size for the three forwarding buffer policies (Figure 4.13), whose reason is that the big buffer can contain more packets.
2) When forwarding buffer size is the same, the average forwarding buffer overflow of No-Split policy is the lowest, and that of Uniform-Split policy is the highest among the three policies.
3) If buffer size is sufficient, the overflow will not occur. In Figure 4.13, when forwarding buffer size is more than 252, the average forwarding buffer overflow of No-Split policy becomes zero; when forwarding buffer size is more than 324, the Flexible-Split policy has no more buffer overflow; when forwarding buffer size is more than 396, the Uniform-Split policy also becomes zero.

- *NAKs*

This parameter refers to the average number of NAKs received by source. In order to avoid the well-known problem of NAK implosion, it is an important criterion to be considered in reliable multicast communication. In Figure 4.14, the tendency of the average number of NAKs for the three policies is similar to the figure for the average forwarding buffer overflow because packet losses are mainly generated by forwarding buffer overflow.

- ***Queue Delay***

As shown in Figures 4.15-4.16, the tendency of average queue delay for the three policies is similar to the average end to end delay.

In conclusion, for small forwarding buffer size (with overflow), we find through the comparison of the above four criteria, that the No-Split forwarding policy is the optimal way to avoid the NAK implosion problem of reliable multicast, with the least NAKs and forwarding buffer overflow, while the Uniform-Split forwarding policy is the worst. But at the same time, we also find that the delay (end to end delay and queue delay) of No-Split policy is the highest among the three policies. Therefore, which policy is more suitable to reliable multicast will be determined by the specific requirement of the different services.

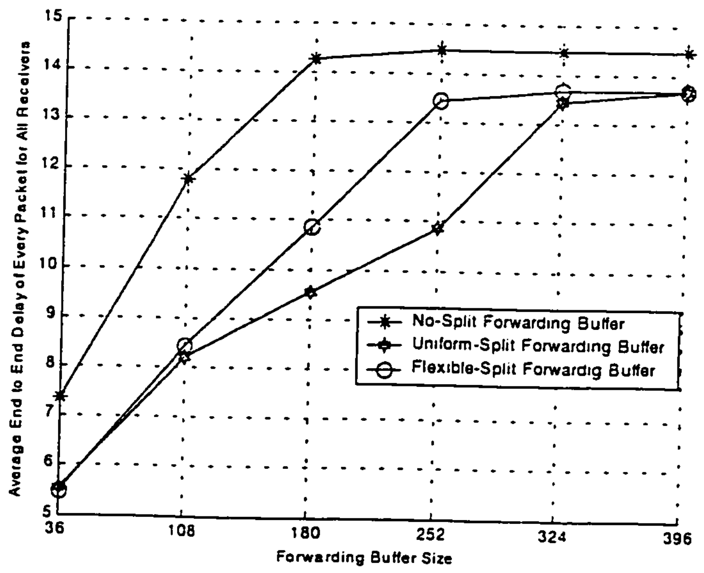


Figure 4.11 Average End to End Delay vs. Forwarding Buffer Size

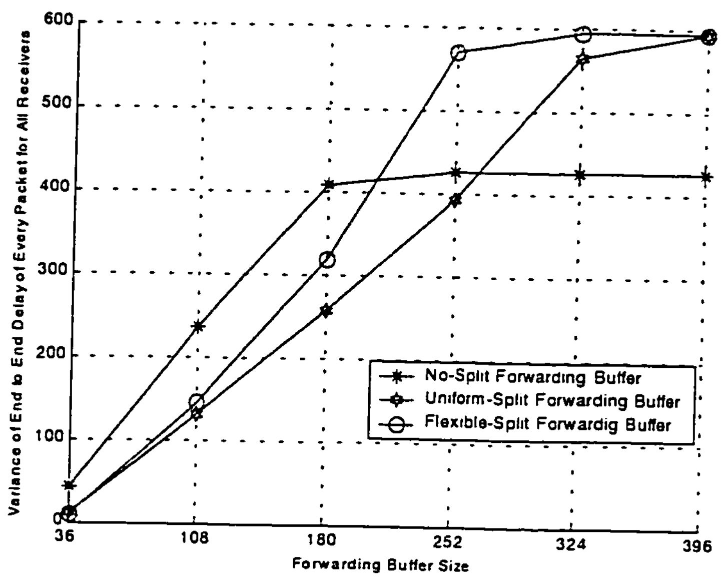


Figure 4.12 Variance of End to End Delay vs. Forwarding Buffer Size

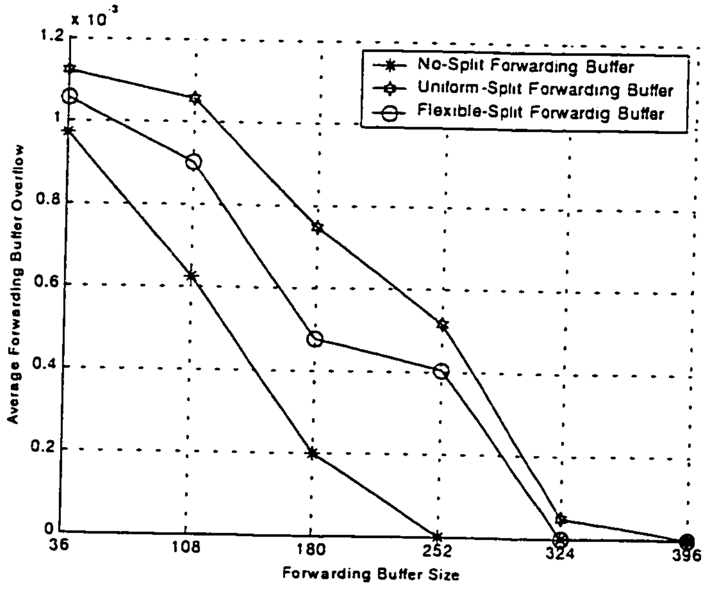


Figure 4.13 Average Forwarding Buffer Overflow vs. Forwarding Buffer Size

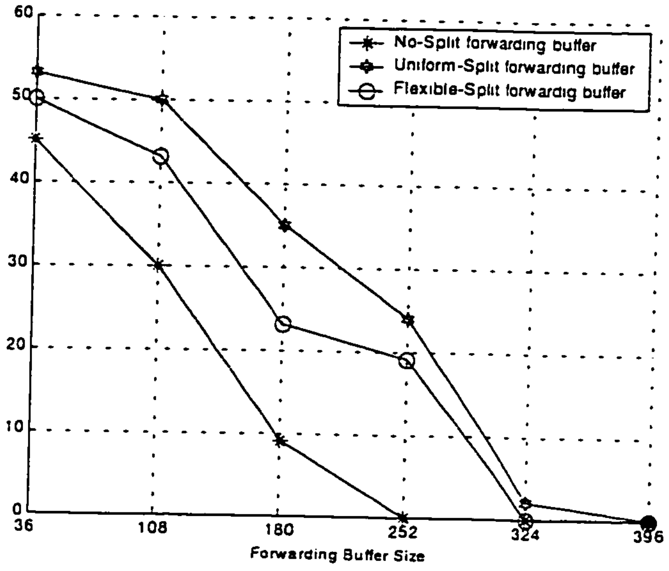


Figure 4.14 Average Number of NAKs vs. Forwarding Buffer Size

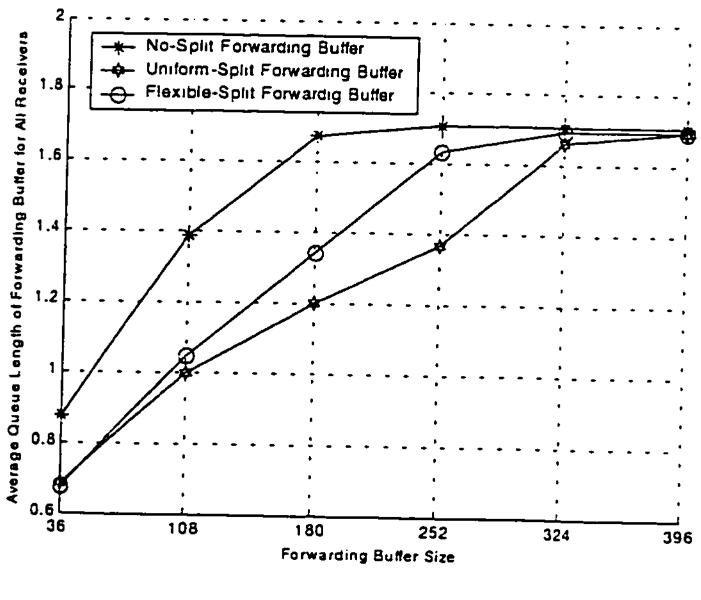


Figure 4.15 Average Queue Length vs. Forwarding Buffer Size

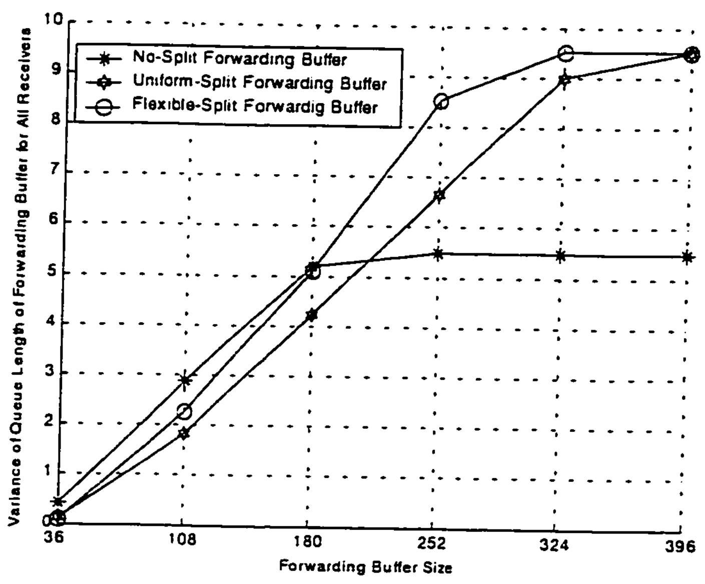


Figure 4.16 Variance of Queue Length vs. Forwarding Buffer Size

4.6.2 Comparison of Three Caching Policies

In the test for the performance of the three caching buffer policies, the Uniform-Split forwarding buffer policy is adopted. The performances of the three policies are explored through 3 criteria: average NAKs received by source, cache hit probability, and average caching buffer overflow by varying caching buffer size, TTL of cached data and packet random loss rate of each link.

I. Vary Caching Buffer Size

Simulations are run by varying the caching buffer size from 6 to 27 and fixing forwarding buffer size=30 packets, TTL of cached data=8 iteration, packet random loss rate of each link=0.01, maximum value of iteration=7000, input traffic load of source 1, 2, 3, 4, and 5 is 0.6, 0.9, 0.3, 0.6, and 0.9.

- **NAKs**

From Figure 4.17, we find that the average NAKs received by source decrease with the increase of caching buffer size until it achieves certain constant value, whose reason is that big-size cache can contain more packets used to loss recovery. When the caching buffer size is less than 18, the NAKs for the No-Split caching buffer policy are less than the other two policies that are almost the same but the Flexible-Split policy is a little bit less. When the caching buffer size is more than 18, the curves of the three policies are almost overlapped with the increasing of caching buffer size.

- **Cache Hit Probability**

From Figure 4.18, we find that the cache hit probability increases with the increase in caching buffer size until it achieves certain maximum value. This can be explained that big-size cache can contain more packets and any lost packets between the caching router and the receiver can be found in the sufficient cache. When the caching buffer size is less than 18, the cache hit probability for the No-Split caching buffer policy are higher than the other two policies, while the other two policies are almost same but the Flexible-Split policy is a little bit higher. When the caching buffer size is more than 18, the curves of the three policies almost overlap with continuously increasing of caching buffer size.

- **Caching Buffer Overflow**

From Figure 4.19, we can get the same conclusion as the NAKs for the three policies. When the caching buffer size increases to a certain value (it is 24 for our case), all data packets can be cached in the caching buffer for a while (TTL of cached data), in other word, the caching buffer overflow becomes zero for all the three polices.

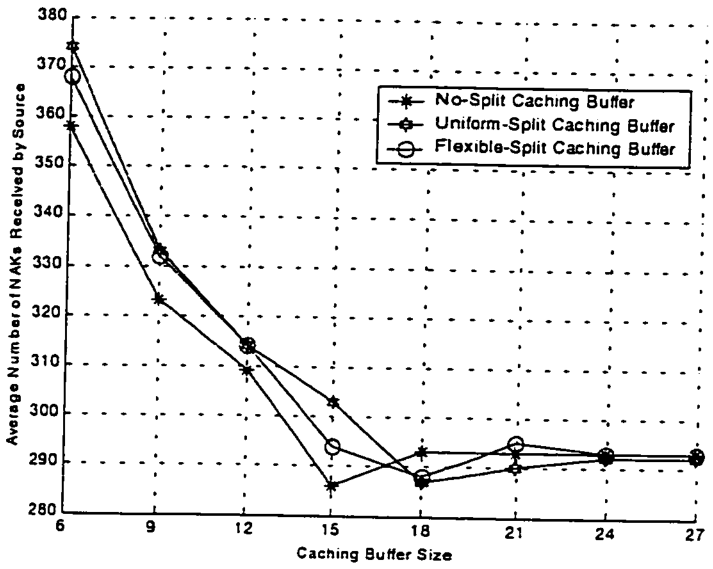


Figure 4.17 Average Number of NAKs vs. Caching Buffer Size

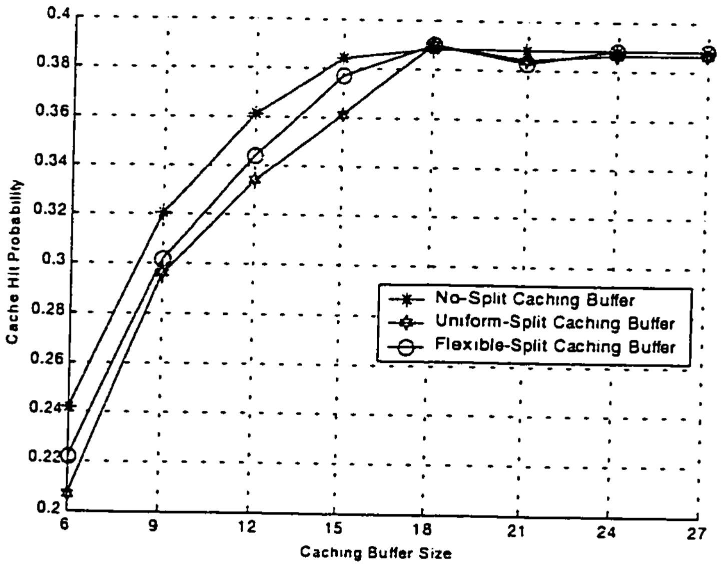


Figure 4.18 Cache Hit Probability vs. Caching Buffer Size

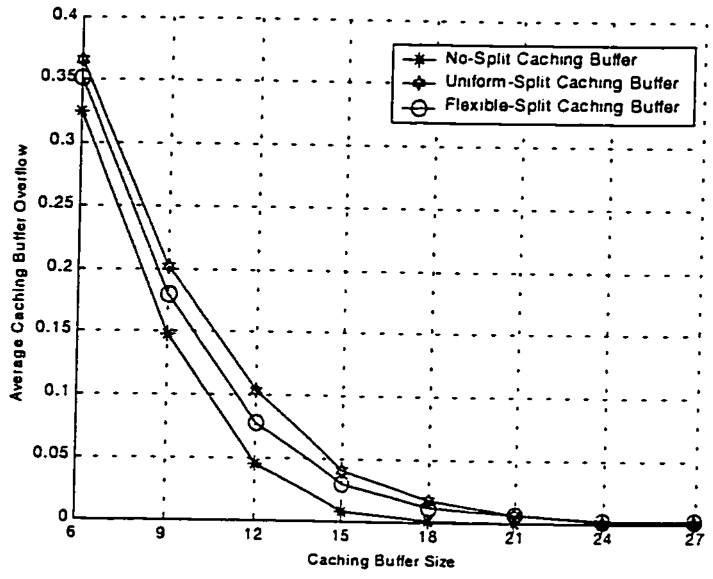


Figure 4.19 Average Caching Buffer Overflow vs. Caching Buffer Size

From the above observation, we can easily draw a conclusion that among the three policies, the No-Split caching buffer policy is the best one due to the relative high cache hit rate, the relative low caching buffer overflow and NAKs.

II. Variation of TTL of Cached Data

Simulations are run by varying the TTL of cached data from 2 iteration to 20 iteration and fixing forwarding buffer size=30 packets, caching buffer size=18 packets, packet random loss rate of each link=0.01, maximum value of iteration=7000, input traffic load of source 1, 2, 3, 4, and 5 is 0.6, 0.9, 0.3, 0.6, and 0.9.

- **NAKs**

From Figure 4.20, we get that the average NAKs received by source for all these three polices can be achieved at the lowest value when TTL is equal to 8, after this lowest value, the number of NAKs will increase with increasing of TTL of caching data. It can be explained that the cache hit probability is highest at TTL=8 (see Figure 4.21), hence more lost packets can be found in caching buffer and the NAKs to source will be decreased accordingly.

- **Cache Hit Probability**

From Figure 4.21, we can find that the performance for cache hit probability is just contrary to the NAKs. The cache hit probability can be achieved at certain peak value which is about 38%, after this peak value, the hit probability will be decreased with the increasing of TTL of caching data.

- **Caching Buffer Overflow**

As for the average caching buffer overflow shown in Figure 4.22, the overflows are increased with the growth of TTL of caching data. The reason for this is that the data packets which are not cached will be more if the TTL is longer. Among these three caching policies, we find that No-Split caching policy provides the least caching buffer overflow. But we should state a point that there is no direct relationship between the caching buffer overflow and the cache hit probability. When the caching buffer size is fixed, the caching buffer overflow can't be too low or too high to attain a relative high cache hit probability.

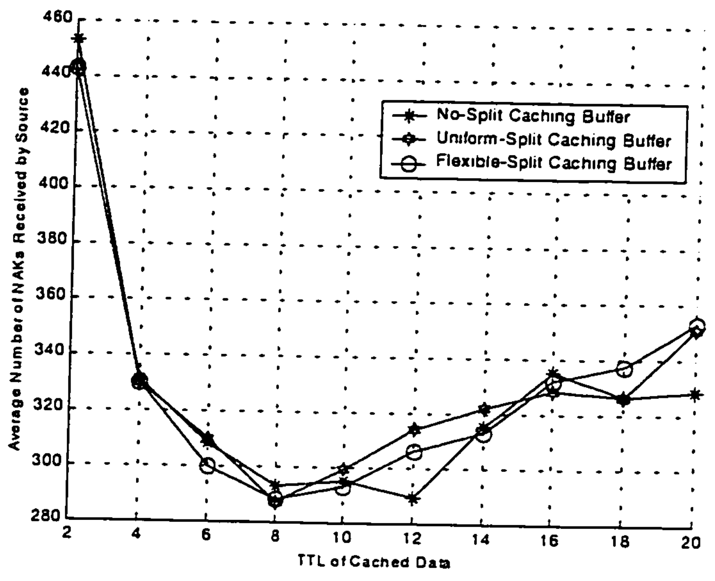


Figure 4.20 Average Number of NAKs vs. TTL of Cached numbers

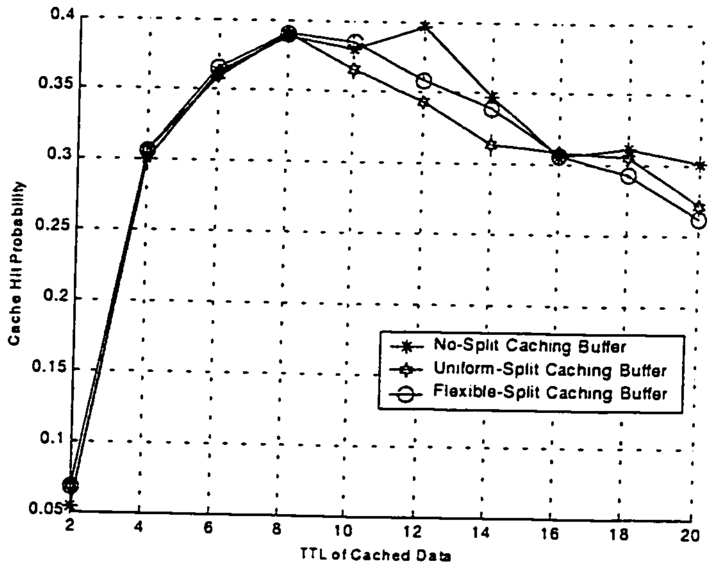


Figure 4.21 Cache Hit Probability vs. TTL of Cached Data

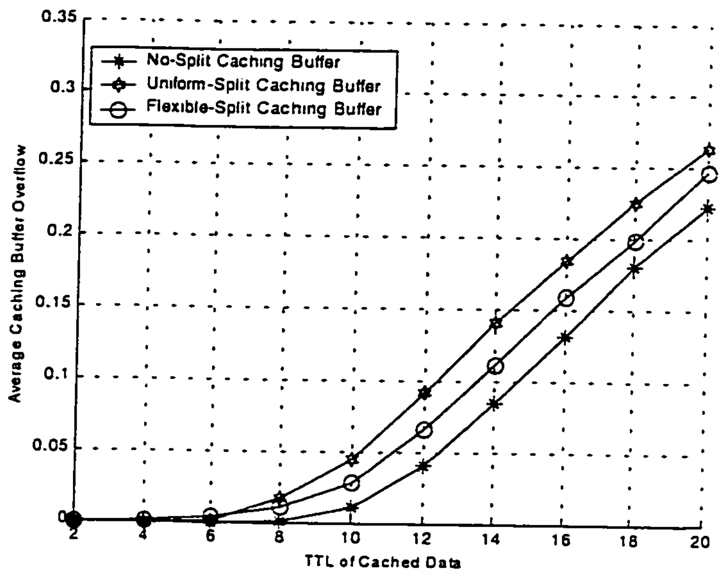


Figure 4.22 Average Caching Buffer Overflow vs. TTL of Cached Data

From the observation of the above three figures, we find that the better performance could be achieved if the TTL value of cached data is properly selected. The optimum value of TTL is mainly determined by the maximum round trip time between this backbone router and the receiver. If TTL is too short, the packet will be discarded before it can be used as retransmission packet. On the other hand, if TTL is too long, it will result in more cache overflows and some packets cannot be cached in this router. In this simulation, the optimal TTL value is equal to 8.

III. Variation of Loss Rate

In order to test this part, we will vary the packet random loss rate of every link from to 0 to 0.02 and fix forwarding buffer size=30 packets, caching buffer size=18 packets, TTL of cached data=8 iteration, maximum value of iteration=7000, input traffic load of source 1, 2, 3, 4, and 5 is 0.6, 0.9, 0.3, 0.6, and 0.9.

From Figure 4.23 and Figure 4.24, we can find that the NAKs or cache hit probabilities of these three policies are almost same. The higher the loss rate increases, the more the NAKs will be received at sources. But the cache hit probability is nearly a constant value when the loss rate is more than zero. This shows that the loss rate has no big effect on the cache hit probability.

In Figure 4.25, the average caching buffer overflow of every policy decreases a little bit with the increase of the loss rate, due to the increase of lost packets on some links before arriving at the caching router. When the loss rate is same, the No-Split caching buffer policy is the lowest among the three policies, while the Uniform-Split caching buffer policy is the highest.

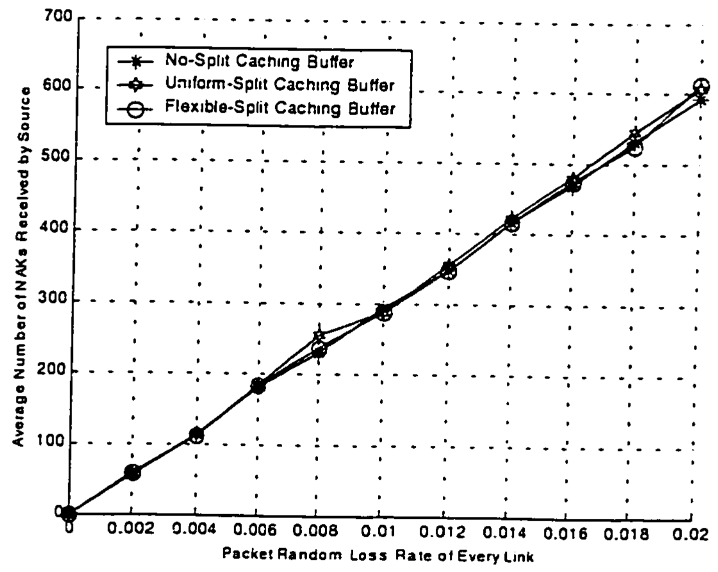


Figure 4.23 Average Number of NAKs vs. Packet Random Loss Rate

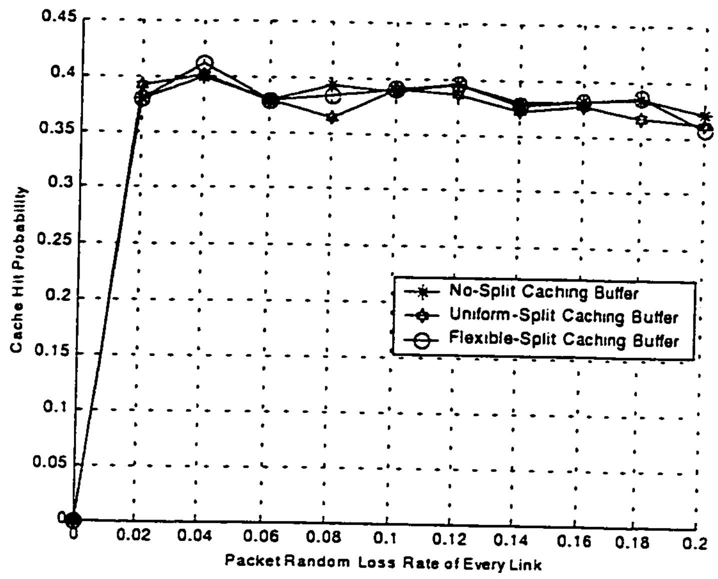


Figure 4.24 Cache Hit Probability vs. Packet Random Loss Rate

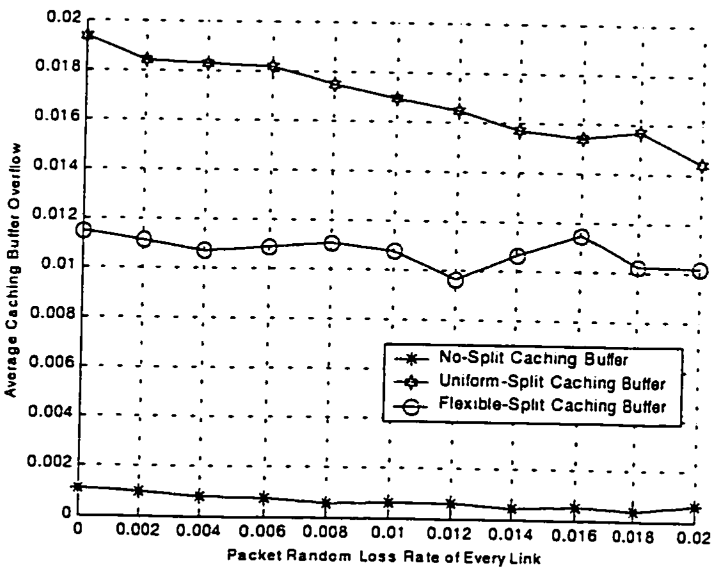


Figure 4.25 Average Caching Buffer Overflow vs. Packet Random Loss Rate

4.6.3 ARQ with Caching vs. ARQ without Caching

For the test of performance of the ARQ scheme with caching, we adopt the following values of the parameters: the forwarding buffer size=30 packets, the caching buffer size=18 packets, the holding time of every cached data=8 iterations, maximum number of iteration=7000. We test two cases that traffic load of each source is 0.2 and 0.6 by varying the packet random loss rate of each link from 0 to 0.02. The forwarding buffer policy is the Unifom-Split policy and the caching buffer policy is the No-Split policy. For the test of perfomance of the ARQ scheme without caching, the two parameters, the caching buffer size and the holding time of cached data (TTL), don't be used, but other parameters are same as the ARQ scheme with caching. We will compare these two schemes from four aspects: end to end delay, NAKs, queue delay, and transmission time of session.

- ***End to End Delay***

Figures 4.26 to 4.29 illustrate the average and variance of end to end delay for different input traffic loads (0.2 and 0.6) when varying the loss rate. The output parameter is calculated by the two different methods mentioned in section 4.4. When input traffic load is fixed (0.2 or 0.6), we observe that the ARQ scheme with caching reduces the end to end delay comparing with the ARQ scheme without caching, and that the end to end delay of each scheme increases with the growth of the loss rate. In addition, for each scheme, the end to end delay for input traffic load of each source=0.6 is greater than that for input traffic load of each source=0.2.

- *NAKs*

The average and variance of NAKs received by source is shown in Figure 4.30 and Figure 4.31.

First, we can observed that for the fixed input traffic load, the higher loss rate leads to more NAKs, and the number of NAKs of the ARQ scheme with caching is distinctly less than that of the ARQ scheme without caching because the caching technique at routers can partly recovery lost data instead of sending their NAKs to the sources. Hence, the caching technique is an important way to solve the NAK implosion problem. Second, whether for the ARQ scheme with caching or for the ARQ without caching, the increasing input traffic load leads to the increasing NAKs even if the loss rate is the same, since the number of lost packets increases with the increasing input traffic load.

- *Queue Delay*

The average and variance of the queue delay are shown in Figures 4.32 to 4.35. When the input traffic loads are fixed, average queue delay increases gradually with the growth of the loss rate, and average queue delay of the ARQ scheme with caching is a little bit less than that of the scheme without caching. The reason is that the growth of the retransmission packets leads to the increase of queuing packets in the forwarding buffer, but some cached packets in the scheme with caching can be used as the repair packets which can reduce certain amount of retransmission traffic passing through some routers. Besides, we can observe that average queue delay for traffic load=0.6 is about 0.28 packet higher than that for traffic load=0.2.

- ***Transmission Time of Session***

Figures 4.36 to 4.37 show the relationship between the average of the transmission time and the loss rate at different input traffic loads. With increasing the packet random loss rate of each link, the transmission time of each scheme will be increased quickly. The transmission time of each scheme at input traffic load=0.6 is higher than that at input traffic load=0.2. Apparently, the scheme with caching improves the performance of multi-session reliable multicast in terms of the session transmission time. For example, when loss rate=0.02 and input traffic loads=0.6 in Figure 4.36, average session transmission time of the scheme with caching is about 5530 iterations, while that of the scheme without caching is about 5860 iterations.

From the above comparison of the four aspects, we could draw conclusion that the ARQ technique with caching are more suitable to reliable multicast in this simulation since it has less delay (end to end delay, and queue delay), less transmission time, and less NAKs.

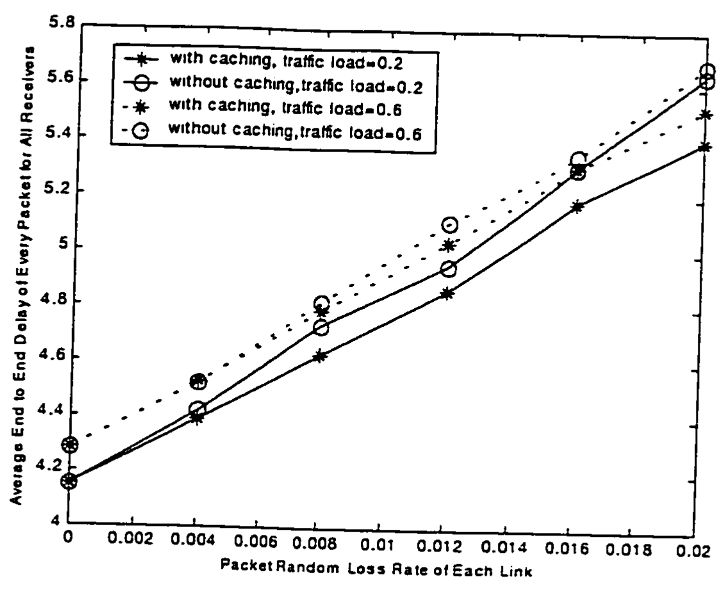


Figure 4.26 Comparison between With Caching to Without Caching: Average End to End Delay for All Receivers vs. Packet Random Loss Rate

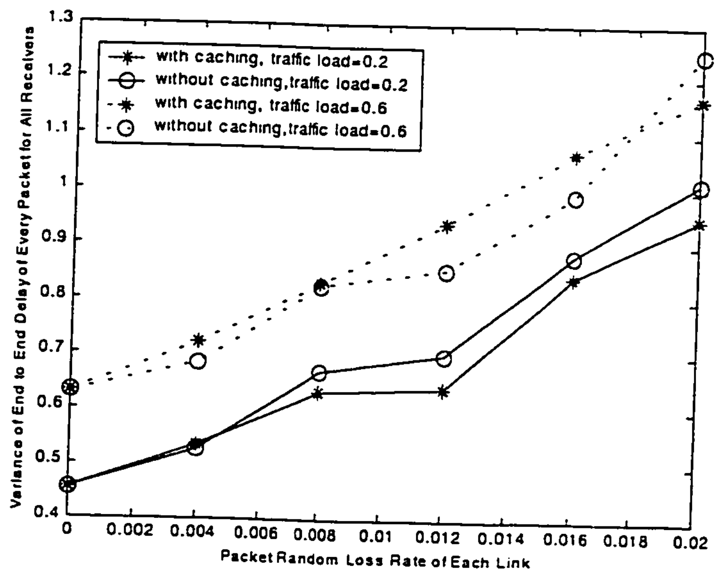


Figure 4.27 Comparison between With Caching to Without Caching: Variance of End to End Delay for All Receivers vs. Packet Random Loss Rate

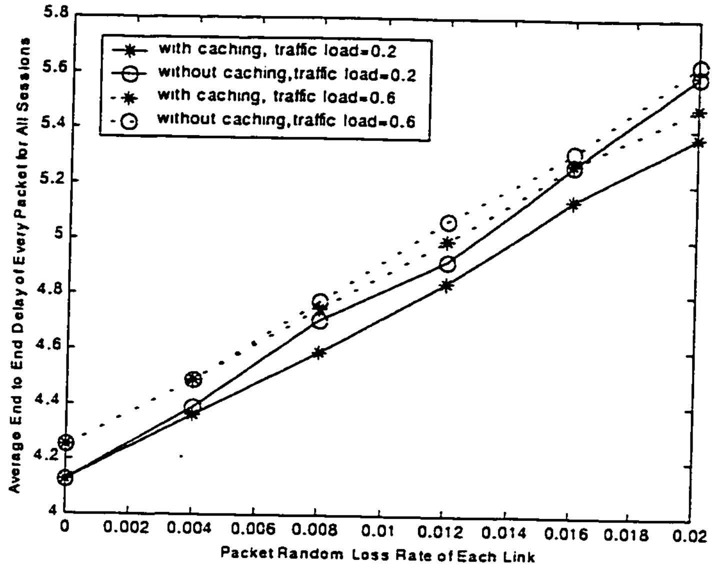


Figure 4.28 Comparison between With Caching to Without Caching: Average End to End Delay for All Sessions vs. Packet Random Loss Rate

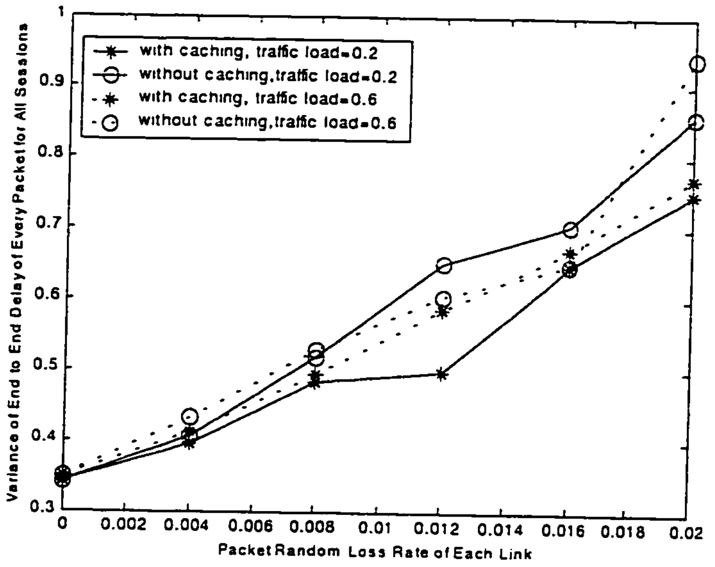


Figure 4.29 Comparison between With Caching to Without Caching: Variance of End to End Delay for All Sessions vs. Packet Random Loss Rate

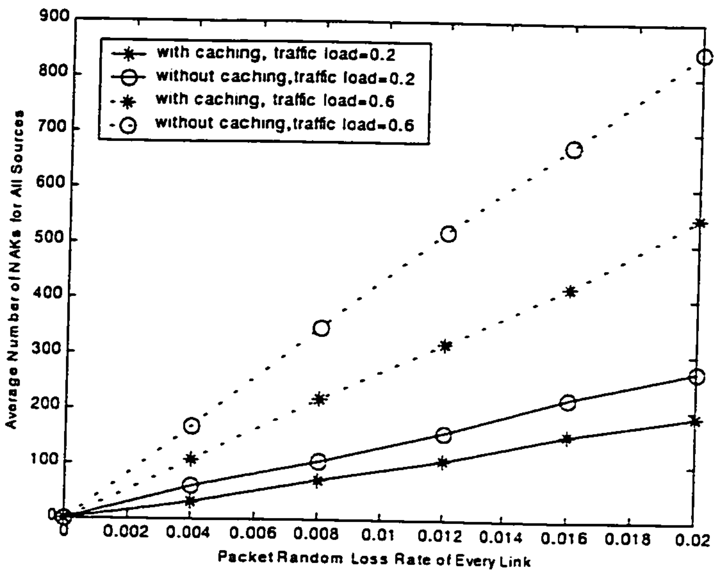


Figure 4.30 Comparison between With Caching to Without Caching: Average Number of NAKs vs. Packet Random Loss Rate

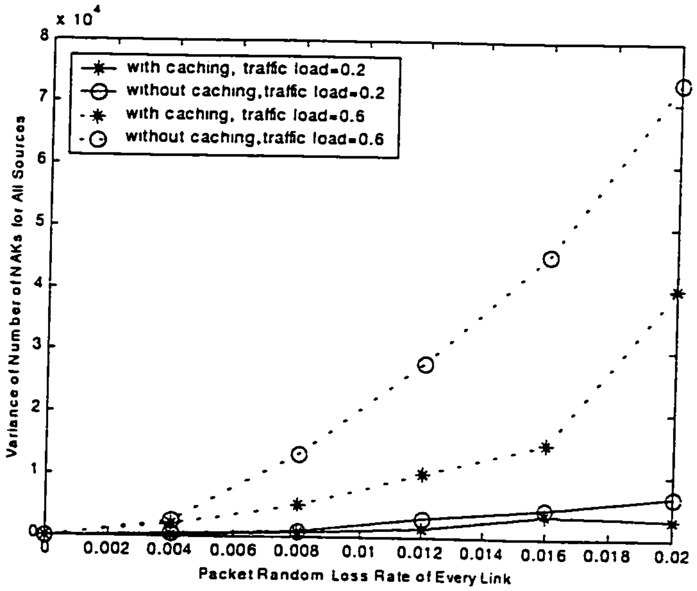


Figure 4.31 Comparison between With Caching to Without Caching: Variance of Number of NAKs vs. Packet Random Loss Rate

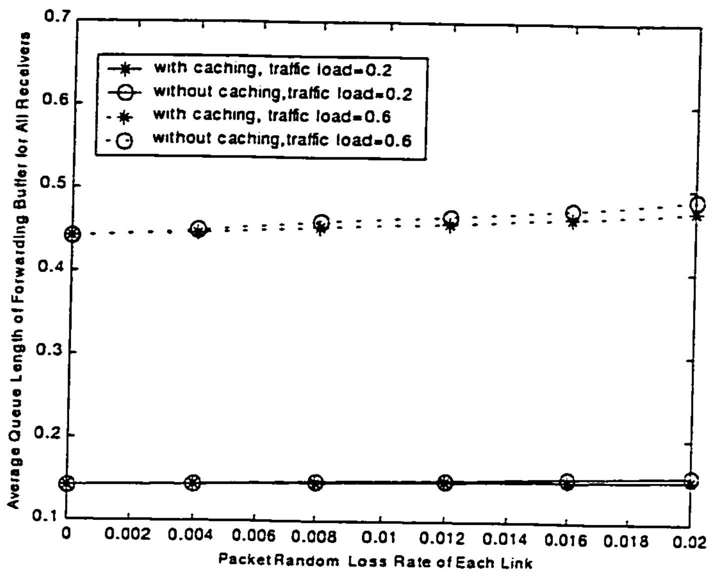


Figure 4.32 Comparison between With Caching to Without Caching: Average Queue Length of Forwarding Buffer for All Receivers vs. Packet Random Loss Rate

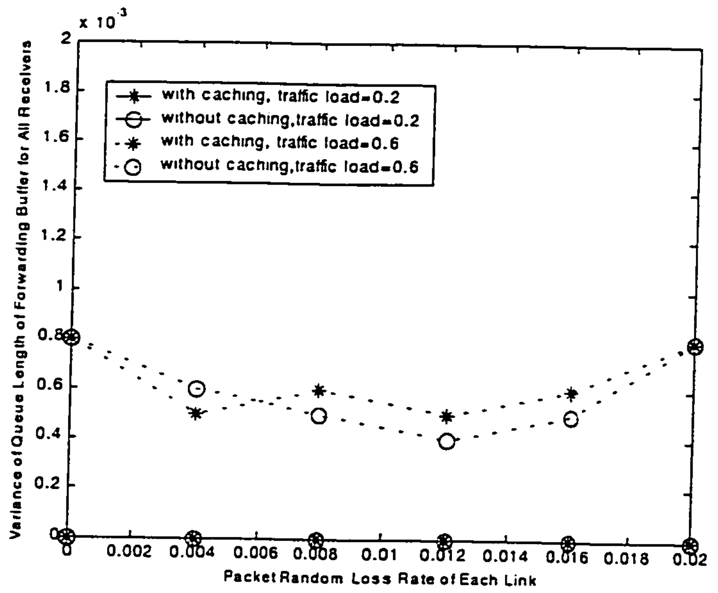


Figure 4.33 Comparison between With Caching to Without Caching: Variance of Queue Length of Forwarding Buffer for All Receivers vs. Packet Random Loss Rate

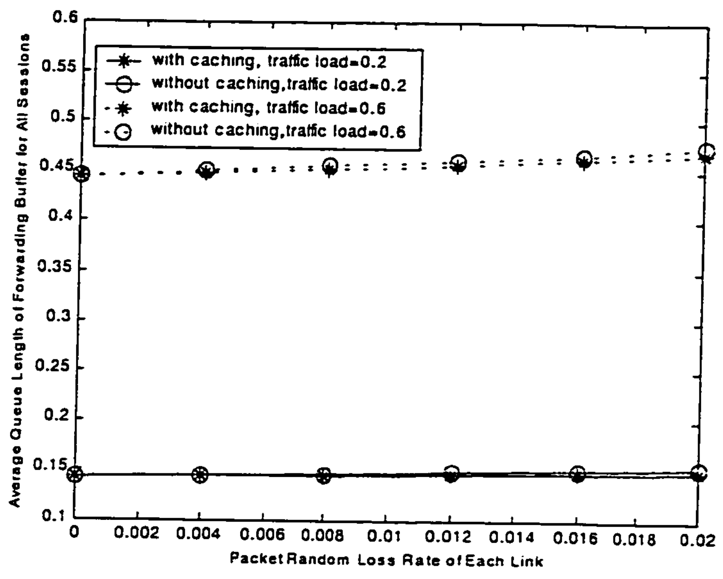


Figure 4.34 Comparison between With Caching to Without Caching: Average Queue Length of Forwarding Buffer for All Sessions vs. Packet Random Loss Rate

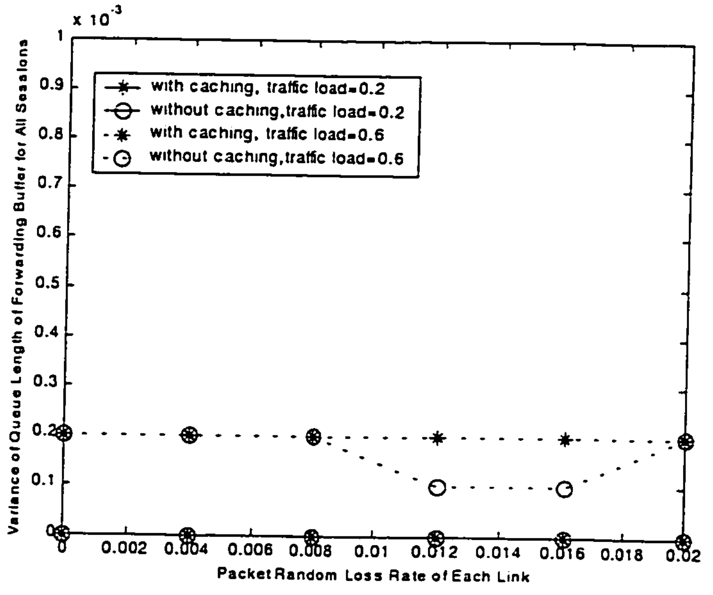


Figure 4.35 Comparison between With Caching to Without Caching: Variance of Queue Length of Forwarding Buffer for All Sessions vs. Packet Random Loss Rate

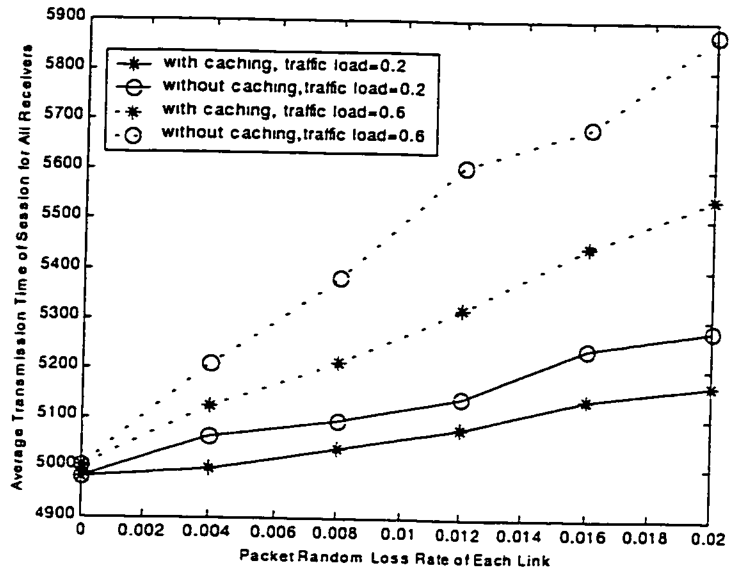


Figure 4.36 Comparison between With Caching and Without Caching: Average Transmission Time of Session for All Receivers vs. Packet Random Loss Rate

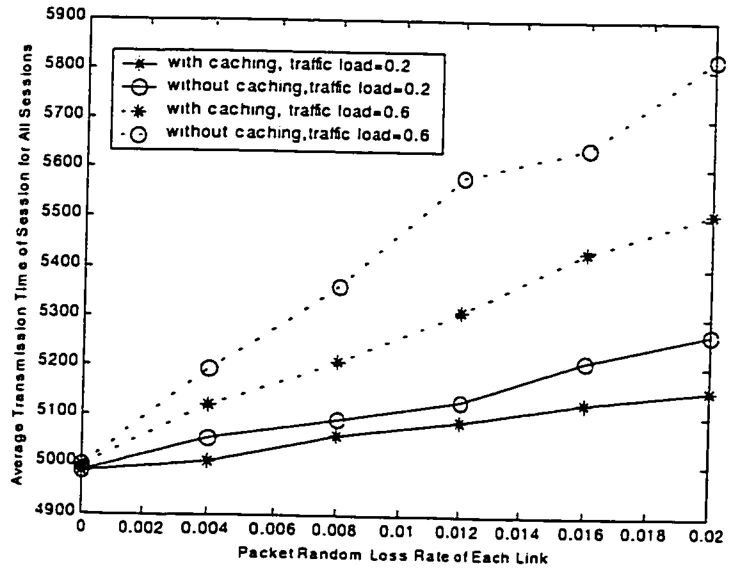


Figure 4.37 Comparison between With Caching and Without Caching: Average Transmission Time of Session for All Sessions vs. Packet Random Loss Rate

4.7 Summary

In this chapter, we have investigated the router buffering and caching techniques for multi-session reliable multicast. We test the performance of different buffering and caching policies of router buffer. Firstly, we introduced some terminologies that will be used in this chapter. Next we gave the simulation model and explained how it works. For the simulation, we selected several meaningful parameters to evaluate the performance, and gave their definitions prior to describe the detailed simulation procedure. Finally, we gave simulation results and analysis in section 4.6.

When we compared three forwarding buffer policies, we found that for small forwarding buffer size (with overflow), the No-Split forwarding policy is a better way to avoid the NAK implosion problem of reliable multicast, with the least NAKs and forwarding buffer overflow, but its delay (end to end delay and queue delay) is the highest among the three policies. Therefore, specific requirement of different services determines which policy should be selected.

As for three cache partitioned policies, we found that at small caching buffer size, the No-Split caching buffer policy is the best one for reliable multicast due to the higher cache hit rate, the lower caching buffer overflow and NAKs.

At last, through the comparison of performance between the ARQ scheme with caching and the ARQ scheme without caching, we conclude that the ARQ scheme with caching improves the performance of reliable multicast in this simulation, which has less delay (end to end delay, and queue delay), less transmission time, and less NAKs.

Chapter 5

Conclusions and Future Work

5.1 Contributions and Conclusions

In this thesis, we have investigated different router buffering and caching policies in our proposed multi-session multicast network. We intended to find a better policy for multi-session reliable multicast by comparing and analyzing the effects of these policies on the performance of reliable multicast through simulation. This is the contribution of our work.

In the first part of our work, we have given three forwarding buffer allocation policies for multiple sessions at routers: No-Split, Uniform-Split, and Flexible-Split. And we have studied the effects of these three policies on the performance of reliable multicast. We have found that at small forwarding buffer size (with buffer overflow), for the amount of the feedback traffic, the No-Split forwarding buffer policy is better than Flexible-Split policy, while the Flexible-Split policy is better than the Uniform-Split policy. But for the latency including transmission delay and queue delay, the case is just the opposite of the above results. Therefore, under our simulation condition, we conclude that the No-Split policy can more efficiently utilize the network bandwidth than other two policies, while the Uniform-Split can improve the latency than the other two policies.

In the second part of our work, we have given three cache partition policies at router and have studied the effects of these three policies on the performance of reliable multicast. We have studied the cache policies from these aspects: cache hit probability, caching buffer overflow, holding time of cached data, and feedback traffic (NAKs) to the source. We have found that for each policy, selecting properly the TTL of cached data, can make this policy to achieve its highest cache hit probability at the same conditions, for example, in this simulation the proper TTL is equal to 8 iterations. At large caching buffer size, the performances of the three polices have not much difference. But at small caching buffer size, the No-Split cache partition policy is superior to other two polices due to the higher cache hit rate, the lower caching buffer overflow, and the lower feedback traffic. While the performance of the Flexible-Split policy and Uniform-Split policy are almost same.

In the last part of our work, we have equipped some routers with the function of caching data for possible loss recovery, and have evaluated the performance improvement of reliable multicast by comparing with the pure ARQ scheme. The scheme of ARQ with router caching data distinctly improved the performance of reliable multicast in this simulation. It reduced latency and session transmission time due to router assistance for loss recovery. Additionally, it saved the network bandwidth due to the reduced feedback traffic to the source and the reduced repair traffic from the source. Of course, the advantage is achieved through the tradeoff between network bandwidth and network-based storage.

5.2 Suggestions for Future Work

What we have done in this thesis is only part of study for this research direction, and there remain many aspects that can be further studied and investigated.

Firstly, we can study these polices by using different network topologies, which will contain more nodes, more sessions, and more links to each receiver.

Secondly, the study of Flexible-Split policy can be further improved by considering more factors, such as the packet arrival process, the number of links to reach a receiver, the link delays, the packets loss probability, etc. Of course, it will be a significantly complicated and difficult work, but it is valuable.

Additionally, unicast data flow and different service class of each session could be considered in the study of the buffer and cache allocation policy, which will make the work more practical.

Furthermore, we could consider having the router suppressing NAKs and subcasting the repair packets to the receivers that request them, just like the active router in ARM.

Finally, we could study these polices at the condition with end to end FEC technique.

References

[1] S.E. Deering, "Host extensions for IP multicasting", RFC 1112, August 1989.

[2] T. Maufer, C.Semeria, 3Com Corporation, " Introduction to IP Multicast Routing", draft- ietf-mboned-intro-multicast-01.txt, March 1997.

[3] C. K. Miller, "Multicast Networking and Applications", (c) 1999 Addison Wesley Longman, Reading MA ISBN 0-201-30979-3.

[4] Douglas E. Comer, "Internetworking with TCP/IP Volume I: Principles. Protocols. and Architecture 3rd ed.", 1998 Prentice Hall, ISBN 7-302-02946-6.

[5] W. Fenner, "Internet group management protocol, version2", Network Working Group, RFC 2236, Nov. 1997.

[6] B. Cain, S. Deering, I. Kouvelas, and A. thyagarajan, "Internet group management protocol, version 3", Internet draft, draft-ietf-idmr-igmp-v3-09.txt, Jan. 2002.

[7] T.Bates *et al.*, "Multiprotocol extensions for BGP-4", Network Working Group, RFC 2283, Feb. 1998.

[8] D. Farinacci *et al.*, " Multicast source discovery protocol (MSDP)", Internet draft, draft-ieft-msdp-spec-13.txt, Nov. 2001.

[9] S. Kumar *et al.*, " The MASC/BGMP architecture for interdomain multicast routing", in *Proc. ACM SIGCOMM*, Vancouver, BC, Canada, Aug./Sept. 1998, pp.93-104.

[10] D. Waitzman, C. Partridge, and S. Deering, "Distance vector multicast routing protocol (DVMRP)", Network Working Group, RFC 1075, Nov. 1988.

[11] J. Moy, "Multicast wxtensions to OSPF", Network Working Group, RFC 1584, Mar. 1994.

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

[13] J. Moy, "OSPF version 2", Network Working Group, RFC2178, Apr. 1998

[14] B. Fenner, M. Handley, H. Holbrook, and J. Kouvelas, "Protocol independent multicast sparse mode (PIM-SM): Protocol specification (revised)", Internet draft, draft-ietf-pim-sm-new-v2-01.txt, Mar. 2002.

[15] A. Ballardie, "Core-based trees (CBT version2) multicast routing", Network Working Group, RFC 2189, Sept. 1997.

[16] A. Mankin *et al.*, "IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols", Network Working Group, RFC 2357, 1998.

[17] J. C. Lin and S. Paul, "RMTP: A reliable multicast transport protocol", in *Proc. IEEE INFOCOM*, San Francisco, CA, Mar. 1996, pp. 1414–1424.

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

[19] H. Holbrook, S. Singhal, D.R. Cheriton, "Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation", *SIGCOMM'95*, Cambridge, MA, USA.

[20] B.N. Levine, J.J. Garcia-Luna-Aceves, "A Comparison of Reliable Multicast Protocols", *Multimedia Systems (ACM/Springer)*, Vol. 6, No.5, August 1998.

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

[22] B. Whetten and G. Taskale, "An overview of reliable multicast transport protocol II", *IEEE Network*, vol. 14, pp. 37–47, Jan./Feb. 2000.

[23] D. Rubenstein, J. Kurose, and D. Towsley, "A Study of Proactive Hybrid FEC/ARQ and scalabel Feedback Techniques for reliable, real time Multicast", *Computer Communication Journal*, Elsevier Publisher, Vol. 24, 2001, pp. 563-574.

[24] S. Bhattacharyya, D. Towsley, and J. Kurose, "The loss path multiplicity problem in multicast congestion control", in *Proc. IEEE INFOCOM*, New York, Mar. 1999, pp. 856–863.

[25] S. Ha, K.-W. Lee, and V. Bharghavan, "A simple mechanism for improving the throughput of reliable multicast", in *Proc. IEEE Int. Conf. Computer Communications and Networks*, Boston, MA, Oct. 1999, pp. 372–377.

[26] T. Jiang, M. H. Ammar, and E. W. Zegura, "Interreceiver fairness: A novel performance measure for multicast ABR sessions", in *Proc. ACM SIGMETRICS*, Madison, WI, June 1998, pp. 202–211.

[27] H.-Y. Tzeng and K.-Y. Siu, "On max–min fair congestion control for multicast ABR service in ATM", *IEEE J. Select. Areas Commun.*, vol. 15, pp. 545–556, Apr. 1997.

[28] J. Mahdavi and S. Floyd. (1997) TCP-friendly unicast rate-based flow control. [Online]. Available: http://www.psc.edu/networking/papers/tcp-friendly.html

[29] S. Lin, D.J. Costello, M.J. Miller, "Automatic-repeat-request error-control schemes", *IEEE communication magazine*, 22(12):5-17, 1984.

[30] D. Towsley, J. Kurose, and S. Pingali, "A comparison of sender-initiated and receiver-initiated reliable multicast protocols", *IEEE J. Select. Areas Commun.*, vol. 15, pp. 398–406, Apr. 1997.

[31] Anthony J. McAuley, "Reliable Broadband Communication Using a Burst Erasure Correcting Code", *ACM SIGCOMM '90*, September 1990.

[32] L. Rizzo, L. Vicisano, "Effective erasure codes for reliable computer communication protocols", *ACM Computer Communication Review*, April 1997.

[33] M. Luby, et al. "Practical Loss-Resilient Codes". *Proceedings of the 29th ACM Symposium on Theory of Computing*, 1997.

[34] R.H. Deng, "Hybrid ARQ Schemes for Point-to-multipoint Communication over Nonstationary Broadcast Channels", *IEEE transactions on communications*, vol. 41, No. 9,September 1993.

[35] J. Nonnenmacher, E. W.Biersack, D. Towsley, "Parity-based loss recovery for reliable multicast transmission", *SIGCOMM'97*, September 1997.

[36] R.G. Kermode, "Scoped Hybrid Automatic Repeat Request with Forward Error Correction (SHARQFEC)", *SIGCOMM'98*, September 1998.

[37] J.J. Metzner, "An improved broadcast retransmission protocol", *IEEE Transactions on Communications*, vol.COM-32, no.6, June 1984.

[38] C. Papadopoulos, G. Parulkar, and G. Varghese, "An error control scheme for large-scale multicast applications", in *Proc. IEEE INFOCOM*, San Francisco, CA, Mar.–Apr. 1998, pp. 1188–1196.

[39] A. M. Costello and S. McCanne, "Search party: Using randomcast for reliable multicast with local recovery", in *Proc. IEEE INFOCOM*, New York, Mar. 1999, pp. 1256–1264.

[40] Y. Gao, Y. Ge, and J. C. Hou, "RMCM: Reliable multicast for corebased multicast trees", in *Proc. IEEE Int. Conf. Network Protocols*, Osaka, Japan, Nov. 2000, pp. 83–94.

[41] L. H. Lehman, S. J. Garland, and D. L. Tennenhouse, "Active reliablemulticast", in *IEEE INFOCOM*, San Francisco, CA, Mar./Apr. 1998, pp. 581–589.

[42] S. K. Kasera *et al.*, "Scalable fair reliable multicast using active services", *IEEE Network*, vol. 14, pp. 48–57, Jan.–/Feb. 2000.

[43] T. Speakman *et al.*, "PGM Reliable Transport Protocol Specification", Internet draft, draft-speakman-pgm-spec-04.txt, 2000.

[44] D. Li and D. R. Cheriton. "OTERS (On-Tree Efficient Recovery using Subcasting): A Reliable Multicast Protocol", *Proceedings of 6th IEEE International Conference on Network Protocols (ICNP'98).* October 1998, Austin, Texas, pp 237-245.

[45] M. Calderon *et al.*, "Active network support for multicast applications", *IEEE Network*, vol. 12, pp. 46–52, May/June 1998.

[46] M. Yajnik, J. Kurose, and D. Towsley, "Packet loss Correlation in the MBone Multicast Network", *IEEE Global Internet mini-conference, GLOBECOM'96, 1996.*