Simulation of Loss Control Technology for Reliable Multicast

Yi Qin

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

August 2002

ABSTRACT

Simulation of Loss Control Technology for Reliable Multicast

Yi Qin

Multicast is positioned to save bandwidth by reducing redundant data and getting the same information to many locations nearly at the same time. This model of Multicast communication is often called "best effort" and corresponds to what the IP network layer provides. A reliable multicast techniques can be added on top of the above best effort service in order to provide lossless and ordered delivery of data to receiving applications

In this thesis, we study the performance of a new loss control technique for reliable multicast. This technique overcomes the effect of packet loss, delay, and jitter in multicast transmission; the performance of the network is highly improved. Simulation results give the characteristics and functionalities of the potential techniques, including average end-to-end delay, end-to-end throughput and average buffer overflow.

ACKNOWLEDGEMENTS

At the outset I would like to express my sincere gratitude to my thesis supervisor, Dr. A.K.Elhakeem, for his guidance, support, encouragement, especially his patience during the entire course of this thesis. This work was born from his ideas and his intellectual properties; also he provided constructive suggestions to enable me to demonstrate this technique analysis and computer simulation. Indeed, I have learned enormously from him for both my career and life during my studies in Concordia University. It is true that "thanks" is just a small word, however, I will for life hold as much appreciation as this nice little word may indicate.

I give my special thanks to Mr. Liu Gang, who give me very useful advice and suggestion to this thesis. Also I would like to thank my great group of friend in Montreal who help me since I come to Canada.

Finally this is dedicated to my wonderful family. They are always standing besides me and helping me with encouragement during the entire period of the study.

TABLE OF CONTENTS

TABLE OF FIGURES

Abbreviations

AAP	Multicast Address Allocation Protocol
ACK	Acknowledgment
ARQ	Automatic Repeat reQuest
BGMP	Border Gateway Multicast Protocol
CRC	Cyclic Redundancy Check
COS	Cost of Service
DR	Domain Receiver
DVMRP	Distance Vector Multicast Routing Protocol
EIGRP	Enhanced Interior Gateway Routing Protocol
FEC	Forward Error Correction
IGMP	Internet Group Management Protocol
IP	Internet Protocol
LAN	Local-Area Network
MADCAP	Multicast Address Dynamic Client Allocation Protocol
MASC	Multicast Address-Set Claim Protocol
MBGP	Multiprotocol BGP (BGP4+)
Mbone	Multicast Backbone
MOSPF	Multicast Extensions to OSPF
MSDP	Multicast Source Discovery Protocol
NAK	Negative Acknowledgment
OSPF	Open Shortest Path First
PIM	Protocol Independent Multicast
RIP	Routing Information Protocol
RTT	Round Trip Time
RS	Reed-Solomon codes
RTCP	Real-Time Control Protocol
RTP	Real-Time Transport Protocol
SAP	Session Announcement Protocol
SDP	Session Description Protocol
TPDU	Transport Protocol Data Unit
UDP	User Datagram Protocol

CHAPTER 1

Introduction

1.1 Motivation

In recent times, more and more real-time multi-point communications essentially require multicast communication. The applications such as defense and intelligence, e-education, and medical imaging transmission are increasing.

Multicast has quickly turned to be a very significant subject in network communication. Much cooperative work has been conducted by various agencies and universities to design supportive mechanisms for multicasting. These mechanisms were referred to as multicast protocols.

Unfortunately, although various multicast protocols currently exist, none of these protocols succeed provide of major reliable functions. For example: "Best effort" transmission can not satisfy reliable requirement in Multicast

ACK implosion at source

Inefficient to have a sender multicast the lost data to the entire multicast tree

Recovery latency

Motivation in this thesis is to introduce a new proposal that could provide major reliable multicast functions such as:

RS coding/ Selective Repeat ARQ/Interleaving

Hybrid FEC/ARQ Used Loss Control

Hybrid Loss Recovery Used

Delay Based Used for Avoidance of NAK Implosion

Hierarchical Organization Used for Reduction in Recovery Latency Recovery Isolation

Adaptation Multicast/ Unicast Retransmission Avoidance Retransmission Implosion

Optimization on DR Location Choose

We are hopeful that this work presented in this thesis will be a step towards achieving a standard reliable multicast protocol.

1. 2 Thesis Organization

The information and work presented in this thesis is organized as follows:

In chapter 1, the major entities of multicast communication are presented. The chapter explores some of the major multicast issues, such as Multicast address, Multicast group, Multicast tree and Multicast protocols.

In chapter 2, the concept of reliable multicasting is explored in depth. In addition, the chapter examines other major multicasting issues such as packet loss, error control types, error recovery types, implosion avoidance, and protocol organization.

In chapter 3, the effects of DR location on QoS performance of reliable multicast are proposed. We give a detailed description of the scheme and its various parameters are presented. We begin

with an analysis of the Average End to End Total Transfer Delay, followed by an evaluation of the Average Buffer Overflow, and verification of End-to-End Throughput of the Network. Results are shown in figures with respect to various parameters. Right through these discussions, we could see the property of simulation hypothesis Network.

In chapter 4, a summary and suggestions for future work are presented.

1.3 Introduction to Multicast Communications

The Internet enables the transmission of a variety of types of data. In particular, there is a growing need to reliably transmit real-time data to numerous receivers where data must be received within a bounded amount of time. Examples range from video and audio communication to reliable delivery of stock data. Using multicast instead of multiple point-to-point unicast connections to distribute this information reduces network bandwidth required to support such applications.

Multicast is positioned to save bandwidth by reducing redundant data and getting the same information to many locations nearly at the same time. Although these goals may sound simple and achievable, they are actually quite difficult to realize.

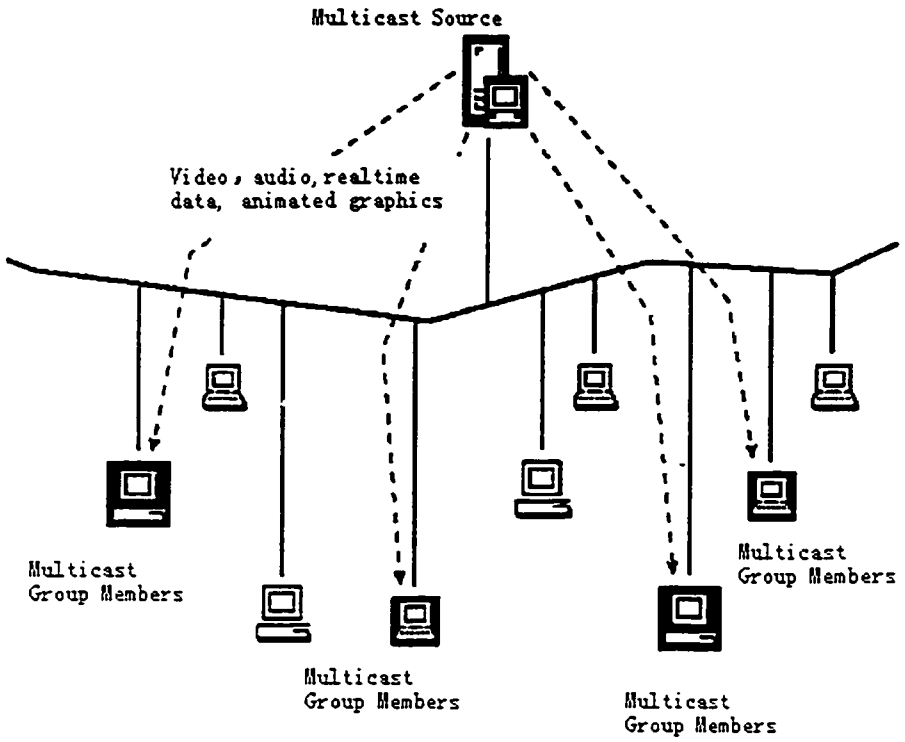


Figure 1-1: Multicast in action

Multicast involves both the maintenance of multicast groups in the LAN and the building and maintaining of trees across LAN subnets and the WAN [31]. Trees are constructed, maintained, and periodically refreshed using multicast protocols. They are stored in routers capable of multicast and can be quite large and memory-intensive. Some multicast protocols use periodic broadcasting to keep trees up-to-date; this practice does not scale well, particularly in the WAN [17]. Change to trees is based on routing table changes and members coming and going from groups. In the network tunnels are used to go across places that do not support multicast; even setting up and maintaining these tunnels can be troublesome at times since both ends must by properly synchronized.

Despite all the current problems with multicast, conserving bandwidth will continue to be increasingly important in our networks. Even with the possibility of practically limitless bandwidth (LAN and perhaps the WAN) the rationale of eliminating redundant transmissions remains. This is especially true with respect to redundant transfer of bandwidth-intensive video across a WAN. As networks become more intelligent and service-rich with QoS guarantees and business levels of COS (Cost of Service) [23], it is expected that we will see an increased interest not only in saving bandwidth, but also in sharing expensive data transmissions that use differentiated services. Clearly it is less expensive to send one express-delivery piece of mail with the exact content from the United States to Austria to fan out to many residents in Tulln, Austria, than to send many separate pieces of express-delivery mail across the Atlantic Ocean [1].

1.1.1 Multicast Addresses

IP multicast uses the Class D IP address range (224.0.0.0 to 239.255.255.255) [4]. This range makes for over 268 million addresses; address ranges 224.0.0.x and 224.0.1.x are reserved for administrative user, and 239.0.0.0 to 239.255.255.255 are reserved for private networks. Although initially this may seem like a lot of addresses, take a moment to consider the considerable number of securities traded across the many stock markets of the world and the need for each to have its own multicast address. You quickly realize that a flat address space of just over a quarter of a billion is less than ideal-perhaps a hierarchical space is in order.

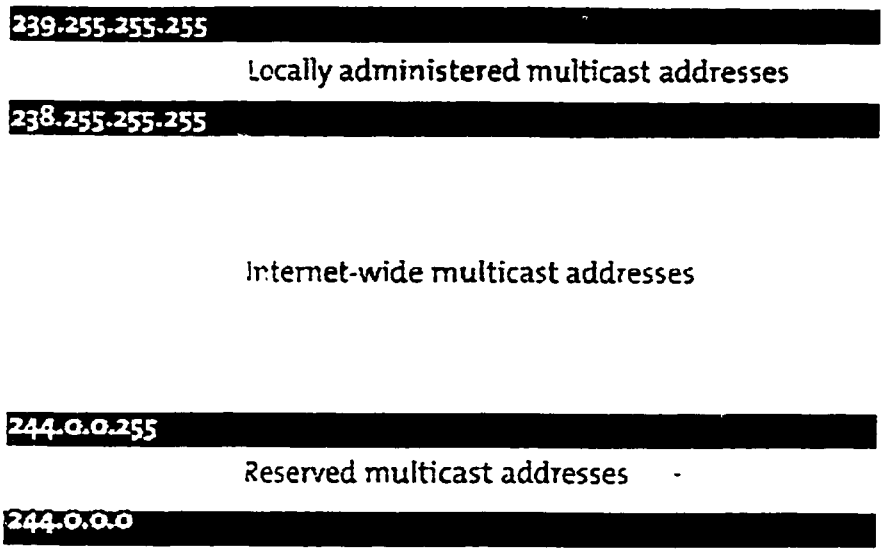


Figure 1-2: Class D map

1.1.2 Multicast Group

IP multicast distribution is based on multicast groups [3]. A group is a collection of one or more senders and receivers, addressed as an entity with a single multicast IP address. This concept enables members to come and go dynamically without the need for senders to adjust the destination address of packets sent. Senders do not have to be members of a group: A group"subscription" indicates that the host is interested in receiving an application's transmissions. Routers keep track of these groups in the form of source/group (where "source" is the source of multicast transmission) pairs and dynamically build distribution "trees" to chart paths from each sender to all receivers. Note that IP multicast facilitates dynamically attaching to a data stream in progress; there is no need to synchronize subscription based on the start or stop of a data stream. Use of groups provides a very powerful abstraction and flexibility for IP multicast.

1.1.3 Internet Group Management Protocol (IGMP)

IGMP is the LAN-based signaling protocol used to manage group membership [1]. One designated multicast router per LAN queries all hosts periodically to refresh group membership and to enable new memberships. End stations belonging to a group respond with a report for each group to which they want to belong. A TTL is set in both queries and reports to limit the scope of the exchange to the local subnet. There are three versions of IGMP [4]; each version extends the protocol's capabilities.

IGMP Snooping is a technique used within switched LAN environments to mitigate the effect of multicast flooding. Essentially IGMP snooping turns off the flooding of switch ports (at layer 2) based on promiscuously monitoring different ports for IGMP messages. On the ports where IGMP traffic is found, IP multicast traffic is forwarded; on the ports without IGMP messages, IP multicast traffic is filtered, greatly reducing the impact of flooding at layer 2 and the potential for congestion that leads to frame dropping.

1.1.4 Expanding-ring search

Expanding-ring search is a technique that leverages the TTL concept in IP packets to find the closest, well known, registered, multicast resource. A sender starts by sending a query with a TTL of 1 to find the closest multicast group of a resource. If no reply is received, the sender keeps incrementing the TTL and making queries until the sender receives the closest reply.

1.1.5 Multicast Tree

When extending IP multicast into the WAN and across large campus networks, multicast trees are formed. Trees are built using a multicast routing protocol-such as Distant Vector

Multicast Routing Protocol (DVMRP) [18] — to link a subnet to all other subnets with members in the same multicast group. There are two types of multicast trees — source based threes and shared trees — and various multicast algorithms associated with each type.

Source-based trees rely on periodic broadcasting and pruning to build and maintain the multicast tree. Multicast packets are periodically broadcast across the network to advertise multicast data. In order to limit broadcasting and reduce duplicate packets, multicast broadcasting is done only for packets that arrive on ports that the router considers the shortest path back to the source of the packet. Edge (or sometimes called leaf) routers use a technique called reverse-path forwarding (RPF) [19] to reduce needless forwarding of multicast packets. If an edge router does not have any subscribers to a multicast stream, it sends a "prune" message back to cut itself off from further messages. The upstream router maintains the" prune state," and after some designated time period, a subsequent broadcast to the edge router is made, enabling the router to graft itself back onto the tree if it obtains an interested subscriber in the meantime. Source-based trees do not scale well in the WAN (broadcast tends to be a bad word when used in the same sentence as WAN), yet they are resilient to network failure since separate trees are maintained for each multicast recipient. Source-based trees are efficient since they follow the shortest path, yet they have the drawback of requiring the maintenance of a lot of individual trees and prune states which use a lot of routing table space that could be used for other purposes like routing tables, and so on.

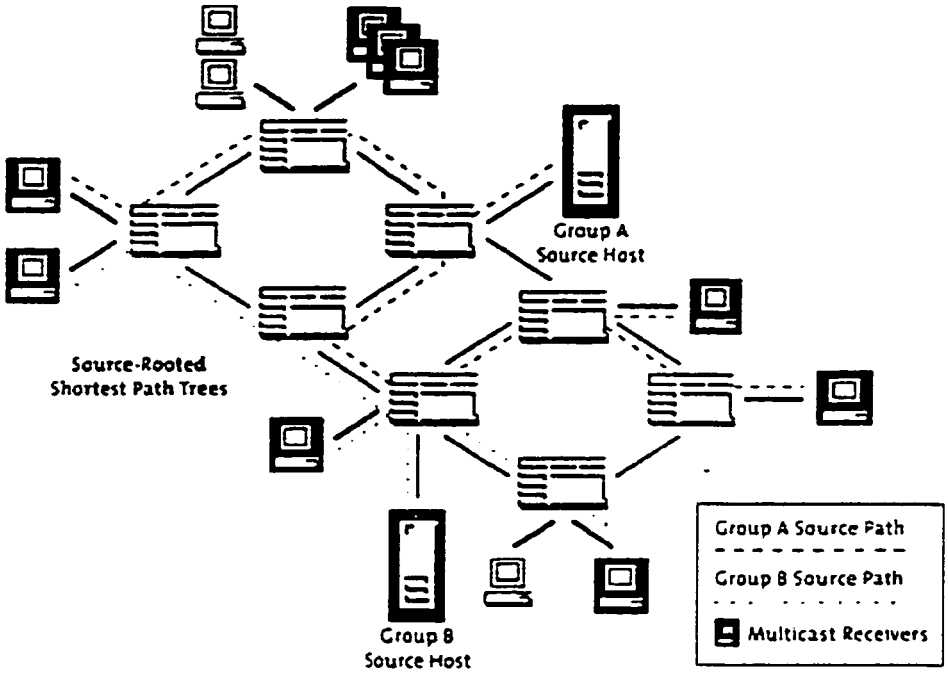


Figure 1-3: Source-Rooted Shortest Path Trees

Shared trees (sometimes called core-based trees) establish in the network a rendezvous point that becomes the center of a multicast group. End stations that want to receive multicasts explicitly for a certain group send a "join" request to the rendezvous point. Also, routes may drop some of these "join" to upstream routes to reduce unnecessary traffic. Core-based trees reduce broadcasting, flooding, and pruning, thus enabling better scale. Core-based trees do, however, result in less efficient paths and are more vulnerable to network failure if linkage to or near the core is severed.

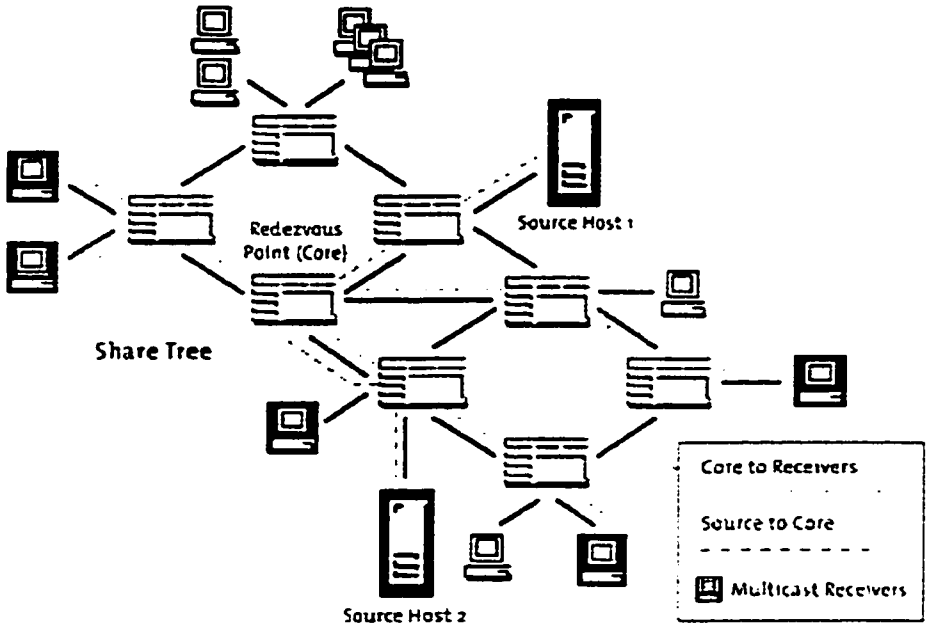


Figure 1-4: Shared Tree

1.1.6 The Mbone and Tunnelling

The Multicast Backbone (MBone) [3] is an experimental collection of multicast router islands that are interconnected by tunneling on top of sections of the internet. DVMRP, MOSPF, and PIM are all used to build the MBone which is designed to test multicast applications before the widespread deployment of multicast routers.

Tunneling is used to get across regions of a network where there are routers that do not support IP multicast traffic. Essentially, a tunnel encapsulates an IP multicast packet into a unicast packet and then "unencapsulates" it at the other end of the tunnel.

1.1.7 Multicast Protocols

Today's Multicast protocol components are shown in Figure 1-5. Multicast protocols are advancing fast .[5]

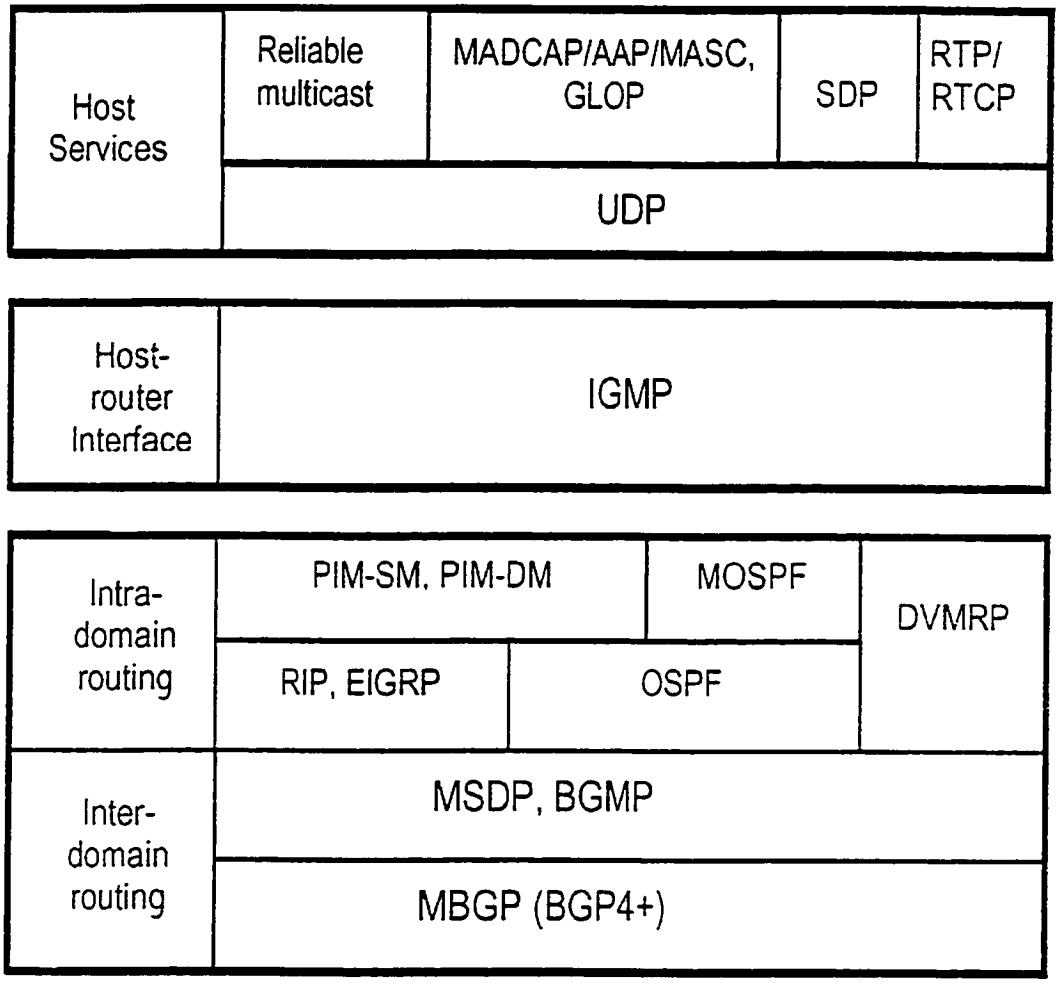


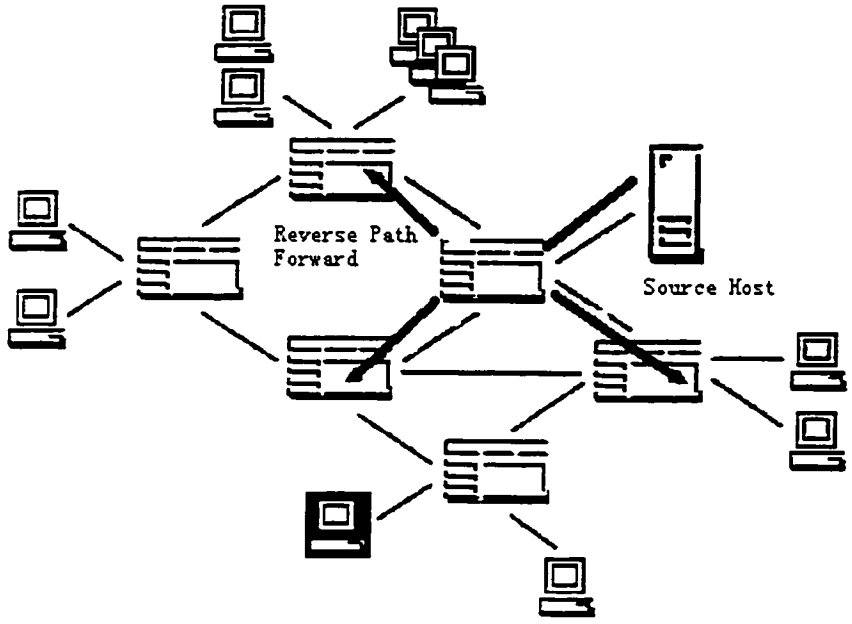
Figure 1-5: Multicast Protocols

1.1.7.1 Distant Vector Multicast Routing Protocol (DVMRP)

DVMRP [17] uses source-rooted shortest path trees as described above. When a source sends its first multicast packet to the group, routers use a technique called Reverse Path Forwarding (RPF) to create the source-rooted tree. With RPF, routers broadcast the initial multicast packet to all interfaces except the one that leads back to the source via the shortest path (see Figure1-6). This

allows multicast frames to reach all subnets without traveling back to the source. Thus the name "reverse path forwarding:" forwarding is based on the router's knowledge of the shortest path back to the source.

Figure 1- 6: Reverse path forwarding



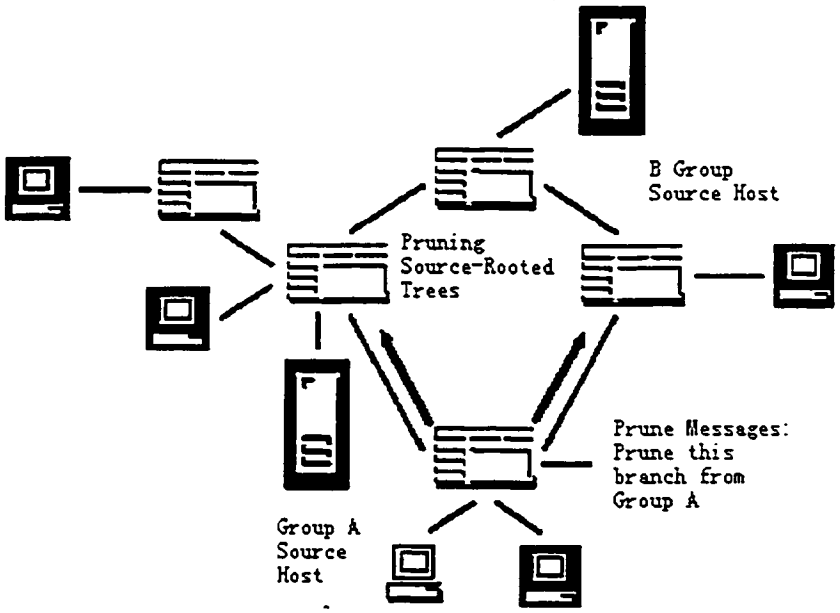
DVMRP routers use a topology protocol similar to Routing Information Protocol (RIP) to compute the shortest path back to the source. The RIP-derived algorithm used by DVMRP computes the shortest path back to the source based on the least number of hops. This is in addition to the unicast routing protocol that all routers run.

As multicast frames are selectively flooded via RPF, they eventually reach all members of the multicast group. During the flooding process, a downstream router may receive a multicast packet addressed to a group that does not reside on any of its interfaces. In this case, the router sends a "prune" message to its upstream neighbors via the reverse path. Prune messages tell upstream

routers not to forward multicast packets down a certain branch of the tree (see Figure 1-7). Prune messages are group-specific, so messages to a branch could be blocked for one group and passed for another. This creates a source-rooted shortest path tree.

Periodically, DVMRP routers allow pruned branches of the tree to "grow back"--to receive multicast messages that were previously blocked. This ensures that a group's multicast traffic will periodically propagate to all corners of the network. A router that receives an IGMP message from a pruned branch will no longer send prune messages upstream for that branch.

Figure 1- 7: Pruning Multicast Delivery Trees



In order to respond quickly to new members, DVMRP routers can also "graft" branches back to the distribution tree without waiting for the prune-aging process. When a router gets an IGMP

message from a receiving host on a pruned branch, it can send a graft message upstream. The graft message cancels any previous prune messages, thereby enabling multicast traffic for the target group to flow immediately to the new member. Graft messages can travel upstream from router to router to allow large pruned sections of a tree to be added back swiftly.

DVMRP is currently the most widely used multicast protocol. It has the advantage of being relatively simple to deploy because it's based on well-known RIP algorithms. Another advantage: the modest processing demands that DVMRP places on routers compared to more sophisticated protocols (however, demands may still be considerable in large applications).

On the downside, DVMRP's reliance on RIP's distance vector routing limits convergence performance and network diameter for large multicast applications. What is more, DVMRP scalability is limited by the need to periodically flood multicast traffic throughout the network. This is impractical when hundreds or thousands of multicast groups share the same network infrastructure.

1.1.7.2 Multicast Open Shortest Path First (MOPSPF)

In spite of the potential for enhancing DVMRP, business clearly needs a more high-performance, scalable approach to multicasting. For many enterprises, the best alternative is MOSPF, a standards-based extension to the popular OSPF routing protocol [17]. OSPF's efficient link state algorithms were developed in response to the insufficiencies of RIP distance vector routines. Likewise, MOSPF multicast delivery represents a substantial improvement over DVMRP.

The IETF OSPF working group under the auspices of RFC 1584 is handling the MOSPF standards process. MOSPF enhances OSPF by letting routers use their link-state databases to build multicast distribution trees for the forwarding of multicast traffic. The standard--defined as an extension to OSPF version 2 --allows MOSPF routers to interoperate with non-multicast OSPF routers for the forwarding of traditional unicast IP traffic.

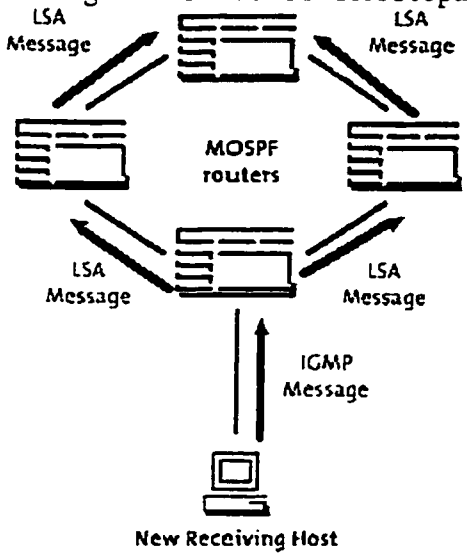
As with DVMRP, MOSPF relies on IGMP as the protocol-receiving hosts use to join multicast groups. A receiving host sends an IGMP message to a local MOSPF designated router (DR) that keeps a database of directly attached members in each group. A backup DR is also defined to perform MOSPF duties in the event that the primary DR fails.

Once an MOSPF DR learns of a new group member on its attached subnets, it sends out a special group membership Link State Advertisement (LSA) message that is propagated to all other MOSPF routers within the OSPF area. When an MOSPF router receives one of these multicast LSAs, it adds the group membership information to its link state database. (This database is the basic OSPF link state database with the MOSPF multicast extensions.) By propagating the locations of multicast group members, MOSPF routers converge to create a detailed map of the multicast topology (see Figure1-8).

When multicast data begins to flow from a source host, the multicast topologies stored in the router's enhanced link-state databases are used to compute forwarding paths. Routers only forward multicast packets to subnets that contain target group members. Subnets without group members are not calculated into the MOSPF multicast tree. This process effectively creates a source-rooted

shortest path tree between each source of the group and its members. As new sources are added to a group, new trees are computed.

Figure 1- 8: MOSPF LSA Propagation



The sophisticated link-state approach of MOSPF allows the multicast distribution system to adapt rapidly to changes in group membership and network resources. If a physical network link goes down, the change is propagated to each router's database to change the multicast route calculation process immediately. Changes are propagated rapidly throughout the network whenever members are added or removed from multicast groups.

MOSPF routers cache these multicast route calculations. Like unicast operations, a multicast forwarding cache eliminates the need to calculate the shortest path for every multicast packet that is passed. In addition, unlike DVMRP, MOSPF can make use of flexible path calculation metrics for

source-rooted path tree construction, going beyond simple hop count. It is important to note that all of this activity is accomplished without the wholesale flooding of multicast packets associated with DVMRP and similar reverse path forwarding techniques.

The previous discussion relates to multicast activities of MOSPF routers within a single OSPF area. Additional aspects of the standard govern how multicast traffic is handled between areas, domains, and autonomous systems, including the traffic moving between single- and multi-domain enterprises. MOSPF defines an Area Border Router function so that a router on the border between two areas can forward multicast traffic bi-directional. This border router forwards a controlled amount of group membership information and multicast traffic between adjoining areas. This controls the extent to which multicast related LSA messages propagate through an enterprise and the Internet. The MOSPF standard also defines a similar role for routers designated as Autonomous System Boundary routers. These devices forward multicast data and control information between Domains and Autonomous Systems.

Finally, MOSPF is "downwardly compatible." It is designed to operate in mixed environments with non-MOSPF routers. This means that migration to MOSPF can be seamless and incremental. During the transition period, MOSPF routers can route multicast traffic around non-compliant routers via a mesh topology that supports its own alternate paths.

1.1.7.3 Protocol Independent Multicast (PIM)

MOSPF and DVMRP are the protocols of choice for enterprises with OSPF backbones and MBONE development. But another multicast technique, Protocol Independent Multicast (PIM)[31], has been proposed as an alternative. PIM has two modalities, PIM Sparse Mode (PIM-SM) and

PIM Dense Mode (PIM-DM).

PIM Dense Mode is targeted at multicast applications with high concentrations of users and high bandwidth capabilities. PIM Sparse Mode is for more thinly populated applications where multicast group members are spread across considerable distance via a variety of transmission types.

PIM is intended to be routing-protocol independent. PIM-DM and PIM-SM therefore do not depend on any specific routing protocol to build and maintain multicast delivery trees. In fact, PIM protocols do not define their own routing update messages in the way that OSPF, RIP, and similar unicast routing protocols do. Rather, PIM relies on the native unicast routing protocol used in the IP network. This requires it to maintain much additional state information to compensate for the fact that the unicast routing table is not multicast-aware.

1.1.7.3.1 PIM-DM

PIM-DM is derived from source-rooted tree methods that create source-rooted trees via Reverse Path Forwarding. PIM-DM, It is similar to DVMRP in the way it floods multicasts out all interfaces except the interface leading back to the source. As with DVMRP, it eliminates unneeded branches with prune messages. Moreover, because it is protocol independent, PIM-DM establishes its own router-to-router dialogues "on top" of the in-place unicast protocol.

PIM-DM is actually somewhat less efficient than DVMRP because it does not control the interfaces that are flooded during the tree building process as tightly as DVMRP. Instead, PIM-DM allows duplicate packets to be flooded as a trade off for a reduced amount of network state

information that is stored by routers. On the other hand, PIM-DM requires less control-processing overhead on routers. This may actually be an advantage for routers with sub-optimal processing architectures. However, most people who are deploying multicast networks today are shying away from PIM-DM. Even the vendors that offer this protocol are encouraging their customers not to implement PIM-DM.

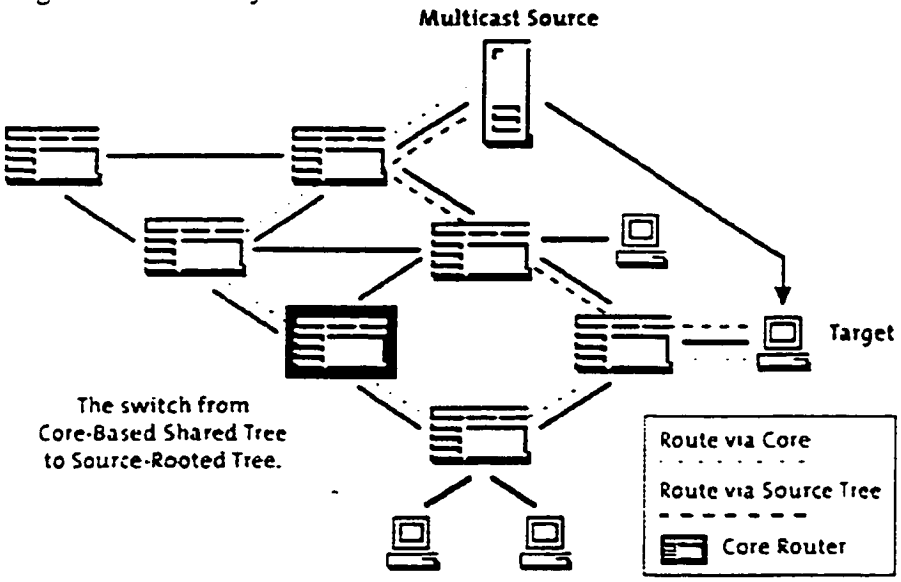
1.1.7.3.2 PIM_SM

PIM's major calling card is its intended ability to support multicast groups spread out across large areas of an enterprise or the Internet. An example of such a group is a videoconference or video broadcast that connects users in many dispersed locations. Another scenario: scattered internetwork users who lack high-bandwidth connections to the corporate backbone. In either case, PIM-Sparse Mode employs shared tree multicasting to avoid the impact of flooding and LSA propagation for very large, dispersed multicast applications.

PIM controls multicast traffic by requiring that multicast routers explicitly join a shared multicast distribution tree. This is the opposite of source-rooted trees, which assume all routers are part of the multicast group until prune messages are generated. A rendezvous point router controls the process of building the PIM-SM shared tree. The process begins when IGMP or similar messages are sent from receiver hosts to local designated routers. DRs in turn signal the rendezvous point (RP) router that joins the DR and its downstream subnets to the tree. The RP serves as a central registry and a conduit for traffic between group sources and receivers. The PIM-SM tree only forwards multicast traffic to branches that have explicitly joined the tree. Hence, the propagation of multicast traffic is tightly controlled.

Although the PIM-SM shared tree avoids wholesale flooding, it subjects multicast traffic to a static, non-optimized set of paths that all pass through the RP router--a potential bottleneck and a single point of failure. To reduce the inefficiencies of the shared tree, the PIM specification gives receivers or routers the option to switch to a source-rooted shortest path tree once the source starts multicasting (see Figure1- 9). This should reduce the non-optimal paths that a central RP introduces into the multicast delivery system. Nevertheless, it also adds greatly to the complexity of multicast protocol deployment: PIM-SM allows a single collection of routers to execute both source-rooted and shared trees in a dynamic, on-demand fashion. But to do so, the single multicast routing protocol must maintain two complete and differing network states.

Figure 1-9: PIM Dynamic Tree Selection



CHAPTER 2

Towards a Reliable Multicast

The degree of reliability required in a multicast communication may vary from application to application. There is an associated cost with achieving reliability, and not all applications are willing to pay for it. At the lowest end of the scale, there is no guarantee of delivery. This model of communication is often called "best effort" and corresponds to what the IP network layer provides. The network drops packets in a silent manner, that is neither sending nor receiving applications are informed about the packet loss. An application relying on a best effort service either does not need reliability or implements the reliability itself, so that error control and recovery mechanisms at lower levels (say, transport-level) are not required or perhaps even desirable. Where to place mechanisms to achieve reliability is a fundamental design issue.

A reliable multicast protocol can be added on top of the above best effort service in order to provide lossless and ordered delivery of data to receiving applications [29]. In such cases, the application at the sending end produces a stream of data, which is to be reproduced at each receiving end. The protocol attempts to hide problems with the transmission of such streams. In

order to hide any data loss by the network, protocol mechanisms incur overhead, which may appear as increased end-to-end delays (loss of throughput) and as additional bandwidth/network cost.

The above scheme guarantees that bytes that are taken by the sending end of the protocol are delivered to the receiving ends (i.e.: to receiving applications) However, it cannot guarantee that the data will be consumed ("read") by the receiving application (the host or application may crash before this happens), neither can it guarantee that once consumed the receiving application will have time to process the bytes. Only an application-level message exchange can provide this guarantee to the sending application, and this is called "end-to-end application reliability". One example of such a communication is when a client sends a request to a server, which sends back a response that works both as an acknowledgment to the receipt of a request as well as the carrier for the results expected from that request. (Note that this is bi-directional communication) The key aspect is that the receiving application only acknowledges the data after it has been safely processed.

In general, any application wishes the data it transmits to be reliably delivered. Some, however, cannot afford to spend time waiting for losses to be recovered; the timely delivery of data is more important. Examples include the broadcast of audio and video over the Mbone. These applications have "soft real-time" requirements: they can sacrifice some of the reliability in favor of speedy delivery of data. As a "live" transmission, the usefulness of data at the receiving applications is time-bounded. For example, if a receiving application employs a "play out buffer" to display a live video stream (with a constant small delay with respect to the actual event), it needs to have the packet available at the moment its information is to be displayed. After that, the packet is of no use.

In conclusion, multicast applications may require different degrees of reliability; in general, the more reliable the service is, the higher is the overhead incurred to achieve it. Some applications

require timely delivery, and thus cannot accept the overhead associated with reliability, With that in mind, applications of reliable multicast have been broadly divided into two categories: timely-delivery and fully-reliable multicast. Fully reliable multicast applications are those where receivers cannot tolerate any losses (e.g.: file transfer). Applications with timely-delivery are those where some degree of loss can be tolerated, but with small end-to-end latency for the data which is successfully delivered. Applications of full-reliable multicast with timely-delivery (i.e.: soft or hard real-time guarantees) are possible, though difficult to realize in the current networks [17].

This thesis focuses on full-reliable (no-real-time) dissemination of data.

2.1 What is reliability?

The definition of reliability has varied widely from one protocol to another. However, a very reasonable definition of a protocol's reliability might be stated as follows: "A protocol is fully reliable if it is capable of providing the needed reliability to a very wide range of applications"[21]. For a protocol to provide this kind of reliability, the protocol must be both sender-reliable as well as receiver-reliable. To achieve that, the sender must be able to have a full knowledge of the receiver set so it can communicate with any receiver to ensure successful delivery of a packet or to initiate an error recovery process. In addition, a receiver must be capable of acknowledging the sender in case it detects an error or a packet loss. With the flat structure, it is quite acceptable to state that the sender has a full knowledge of the receiver set, since there is a direct connection between the sender and the receivers. However, with the hierarchical structure, since there is no direct connection between the sender and the receivers, as receivers are controlled and served by other entities, it is too difficult for the sender to have and maintain a full knowledge of the receiver set.

2.2 Major Reliable multicast issues

2.2.1Packet loss in multicast

A critical issue for reliable multicast protocols is the manner in which packet losses occur in the network. First consider a simple multicast network model where each of the (say, N) receivers maintains an independent, individual channel (C_i) with the sender. Each channel (C_i) has a given non-nil loss rate (ε_i) the probability that a packet sent by one end reaches corrupted or does not reach at all the other end of the channel is (i), When a packet is multicast by the sender, a copy of the packet is sent to each of the channels, any feedback packets that result from receivers go through the same channel from which the original data packet arrived. Figure 2.1 illustrates this arrangement for N = 6[10].

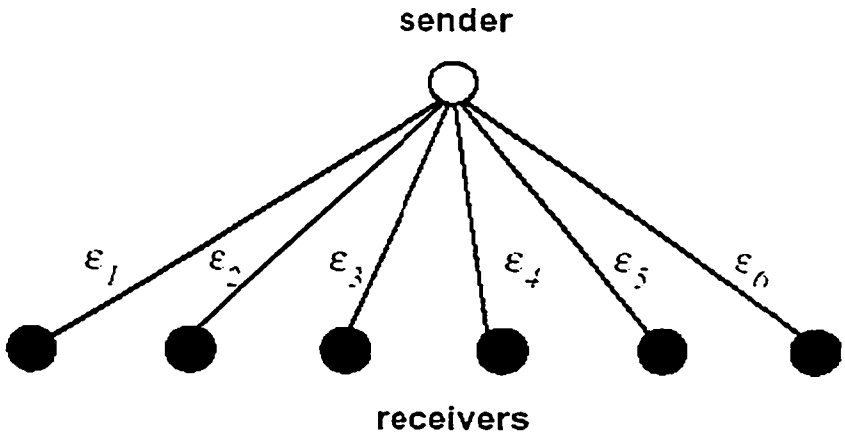


Figure 2-1: Simple Network loss abstraction

Note that if a single copy of the packet is lost in one of the channels, recovery-related action will be required; for most protocols, it means that the sender will have to detect the loss (e.g.,

through feedback from receivers) and it will have to retransmit the lost packet. The probability is that at least one copy of the packet is lost increases with N. So, if N tends to infinity, the probability a given multicast transmission will require recovery tends to 1. In other words, there will be a group size sufficiently large to cause all multicast packets to require recovery by one or more receivers.

The above observation does not address the issue of how many receivers will require recovery of a given data packet. In this simple network model, channels are independent, and so are losses. The number of receivers experiencing a given loss will depend on the loss rates set for individual channels; for example, if two receivers are at the end of very loss channels then the chance that both receivers miss a given packet is, compared to other receivers, higher.

When multicasting in actual networks, instead of independent channels, packets are propagated from sender to receivers through a multicast routing tree. In the Internet, anecdotal evidence suggests that most losses are caused by congestion, that is, by buffer overflow at (packet-switching) routers, not by packet corruption. Even when there is a low probability of loss in each of the nodes of the tree, or in the physical links that connect such nodes, the cumulative loss probability seen by the source at the root of the tree may be quite high.

Note that a packet, which is lost at a given node of the tree, will not arrive at any receiver that is downstream of the point of loss. In the worst case, a packet is dropped at the root itself (before being transmitted) and is therefore missed by all receivers. Hence, the higher the multicast tree is the more overlap exists between paths, and so higher is the probability that a given loss will be correlated among receivers. This loss correlation is spatial: if a given receiver experienced a loss, it is likely that nearby node (siblings and downstream nodes in the tree will experience the same loss. Two examples of losses in a multicast transmission are shown in Figure 2.2; routers are represented

as squares, and receivers as circles; the receivers affected by a loss are marked with an external dashed line. The loss on the left is higher than in the tree on the right, and affects 3 receivers; the one on the right affects a single receiver.

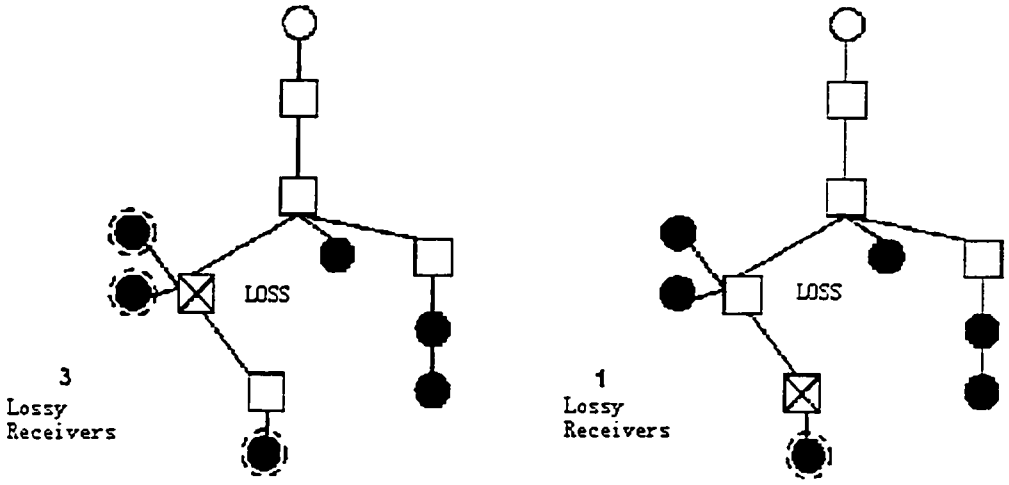


Figure 2-2: Two losses with different "degrees" of spatial correlation

There is a second kind of loss correlation, which is directly related to network congestion: temporal loss correlation. When a router receives more packets than it can forward, it eventually starts dropping packets due to buffer overflow. When this situation arises, it tends to cause several losses: a congested router is likely to drop several packets until the congestion control in the protocols with flows passing through the router react and slow down. Therefore, if a receiver misses a given packet, the chance that one or more of the next few packets will also be missed is greater than the original loss probability.

The experimental study performed in the Mbone and presented in [3] confirms the statements above. It reports that

(a) 47 percent of multicast transmissions required recovery;

(b) Most receivers experienced very low loss rates throughout the transmission;

(c) There were loss bursts: a large number of consecutive packets were lost by a receiver.

However, the study also found that spatial correlation was generally small apart from losses near the root. This is only due to the topology (the Mbone) employed in the experiment, as the authors found that links connecting leaf nodes to the tree ("tail circuits") had higher loss rates than those in the Mbone backbone. LAN bandwidth and resources are likely to stay cheap and plentiful, while tail circuits linking LANs(local-area networks) to WANs(wide area networks) are likely to become more congested in the future. So, more losses are to be experienced either between the root and the backbone (all receivers miss packet) or between the backbone and a receiver (only one or two receivers miss packet). Therefore, generally, a designer of a reliable multicast protocol may expect to find strong temporal loss correlation and possibly also strong spatial loss correlation.

2.2.2 Types of Error Control

Depending on line and network characteristics, transmitted data may be reordered, distorted or deleted and occasionally, duplicated. Transmission errors may be divided in two categories [13]:

- data corruption: distortion or insertion of message contents, and
- data sequencing problems: deletion, duplication, and reordering of messages.

If data can be corrupted, then it is necessary to detect these errors by adding some kind of redundancy to messages, whose consistency is checked at receiver side. Assuming that a corrupted packet will be detected and discarded by the underlying layer, there are two basic methods of error control:

· Forward Error Control (or forward error correction, FEC) [12]: the sender generates redundant codes from the data transmitted, and sends these codes through additional packets (typically at the end). The receiver detects the losses and may be able to reconstruct missing data using the coded data in conjunction with part of the original data which was successfully received.

· Feedback Error Control: the sender detects losses through feedback sent by receivers (NAK), or the absence of it (timeout for ACK), and reacts by retransmit ting missing data.

2.2.2.1 Forward Error Control (FEC)

The idea of FEC is to transmit original data together with some "parities" to allow reconstruction of lost packets at the receiver. The redundant data is derived from the original data using techniques from coding theory [9,13]. The data stream is transformed in such a way that reconstruction of a data object does not depend on the reception of specific data packets, but only on the number of different packets received. There are multiple benefits to using the parity for loss recovery instead of retransmitting the lost packets:

Improved transmission efficiency: A single parity packet can be used to repair the loss of any one of the data packets. This means that a single parity packet can repair the loss of different data packets at different receivers. Improved scalability in terms of group size: When a lost packet is retransmitted via multicast, the packet will be received more than once by all of the receivers that have already successfully received the packet. Such duplicate packets waste transmission bandwidth and processing capacity. Using FEC can significantly reduce the necessity for retransmission requests or make them totally unnecessary, which is also important in delay-sensitive multimedia applications [14, 15].

Another point on FEC error control scheme is its wide availability: FEC implemented in software is reasonably fast on today's desktop PCs.

Reed Solomon codes are a subset of BCH codes and are linear block codes. A Reed-Solomon code is specified as RS(n,k) with s-bit symbols. This means that the encoder takes k data symbols of s bits each and adds parity symbols to make an n symbol codeword. There are n-k parity symbols of s bits each. A Reed-Solomon decoder can correct up to t symbols that contain errors in a codeword, where $2t = n$-k.

The following diagram shows a typical Reed-Solomon codeword (this is known as a Systematic code because the data is left unchanged and the parity symbols are appended):

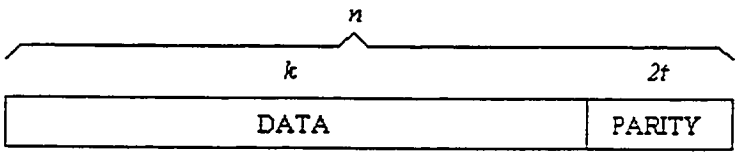


Figure 2-3: Reed-Solomon Codeword

Example: A popular Reed-Solomon code is RS(255,223) with 8-bit symbols. Each codeword contains 255 code word bytes, of which 223 bytes are data and 32 bytes are parity. For this code:

n = 255, k = 223, s = 8

2t = 32, t = 16

The decoder can correct any 16 symbol-errors in the code word: i.e. errors in up to 16 bytes anywhere in the codeword can be automatically corrected.

Code type	RS block code
Symbol size	8 Bits
Codeword size (n)	255 Bytes
Data bytes per codeword (k)	223 Bytes
Parity bytes per codeword(r)	32 Bytes
Maximum burst error correction capability	128 Bits (r·2)
Maximum erasure error recover capability	32 Bytes (r)

Figure 2-4: RS(255,233) codes

Given a symbol size s, the maximum codeword length (n) for a Reed-Solomon code is

$n = 2s - 1$

The amount of processing "power" required to encode and decode Reed-Solomon codes is related to the number of parity symbols per codeword. A large value of t means that a large number of errors can be corrected but requires more computational power than a small value of t.

Symbol Errors: One symbol error occurs when 1 bit in a symbol is wrong or when all the bits in a symbol are wrong.

Reed-Solomon codes are particularly well suited to correcting burst errors (where a series of bits in the codeword are received in error).

Decoding

Reed-Solomon algebraic decoding procedures can correct errors and erasures. An erasure occurs when the position of an erred symbol is known. A decoder can correct up to t errors or up to 2t erasures. The demodulator can often supply erasure information in a digital

communication system, i.e. the demodulator "flags" received symbols that are likely to contain errors.

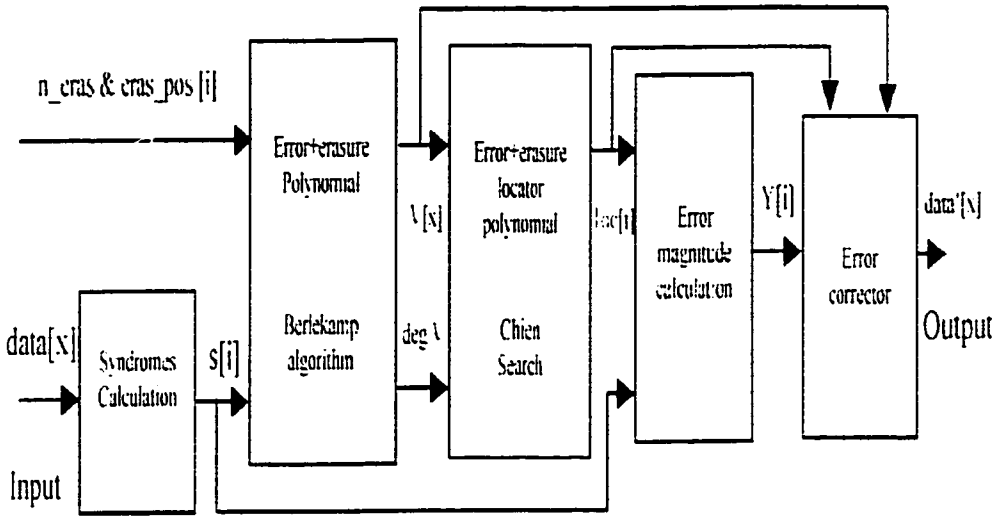


Figure 2-5: RS Decoding Block Diagram

When a codeword is decoded, there are three possible outcomes:

1. If $2s + r < 2t$ (s errors, r erasures) then the original transmitted code word will always be recovered,

OTHERWISE

2. The decoder will detect that it cannot recover the original code word and indicate this fact.

OR

3. The decoder will mis-decode and recover an incorrect code word without any indication.

The probability of each of the three possibilities depends on the particular Reed-Solomon code and on the number and distribution of errors.

2.2.2.2 Automatic Repeat reQuest (ARQ)

As only feedback error control techniques are considered. Deletion (loss), duplication and reordering of packets have to be detected and then recovered by means of retransmissions. This technique is known as automatic repeat request, or ARQ [13].

Four types of ARQ are common:

- Stop and Wait (the simplest) Stop and Wait transmission is the simplest reliability technique and is adequate for a very simple communications protocol. A stop and wait protocol transmits a Protocol Data Unit (PDU) of information and then waits for a response. The receiver receives each PDU and sends an Acknowledgement (ACK) PDU if a data PDU is received correctly, and a Negative Acknowledgement (NACK) PDU if the data was not received. In practice, the receiver may not be able to reliably identify whether a PDU has been received, and the transmitter will usually also need to implement a timer to recover from the condition where the receiver does not respond.

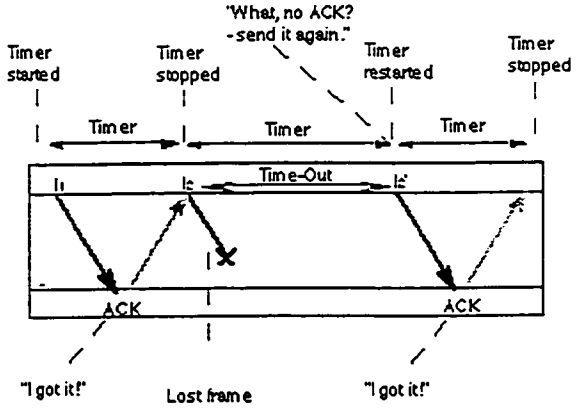


Figure 2-6: Stop and Wait ARQ - Retransmission due to timer expiry

-Polling (sometimes known as "CheckPointing") Polling is a technique where by a master station identifies the status of the slave stations by asking each in turn for a status report. In many cases, communications links consist of just two nodes - so there is just one master station and one

slave station. During normal operation, the sending node to the corresponding receiver transmits a steady flow of PDUs. This receiver returns information (acknowledgments) to the transmitter to synchronise the operation of the protocols (or to indicate that a receiver is 'not ready' to receive data). If a receiver fails to receive an acknowledgment for the PDUs it has transmitted, or when it has sent a PDU which requires the remote node to perform a specific action and this action has not been performed, it may commence a process known as "Polling".

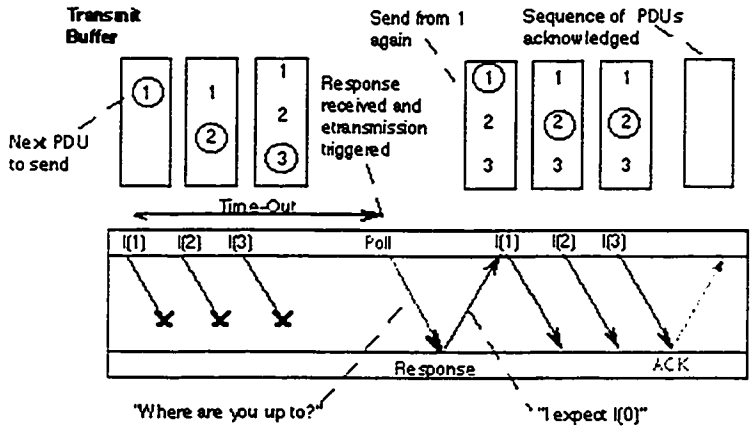


Figure 2-7: Polling ARQ

·Go-Back-N (sometimes known as "Reject") The recovery of a corrupted PDU proceeds in three stages:

First, the corrupted PDU is discarded at the remote node's receiver.

Second, the remote node requests retransmission of the missing PDU using a control PDU (sometimes called a NACK or REJECT). The receiver discards all PDUs which do not have the number of the requested PDU.

The final stage consists of retransmission of the lost PDU(s).

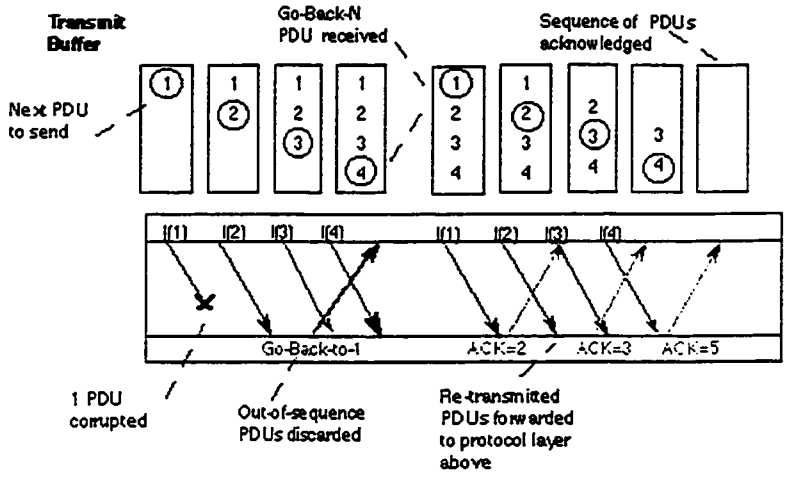


Figure 2-8: Go back N ARQ

·Selective Repeat (sometimes known as "Selective Reject) The recovery of a corrupted PDU proceeds in four stages:

First, the corrupted PDU is discarded at the remote node's receiver.

Second, the remote node requests retransmission of the missing PDU using a control PDU (sometimes called a Selective Reject). The receiver then stores all out-of-sequence PDUs in the receive buffer until the requested PDU has been retransmitted.

The sender receives the retransmission request and then transmits the lost PDU(s).

The receiver forwards the retransmitted PDU, and all subsequent in-sequence PDUs which are held in the receive buffer

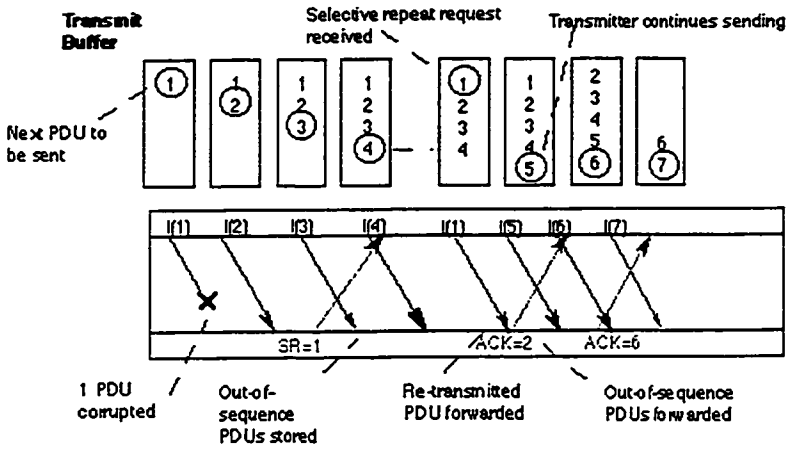
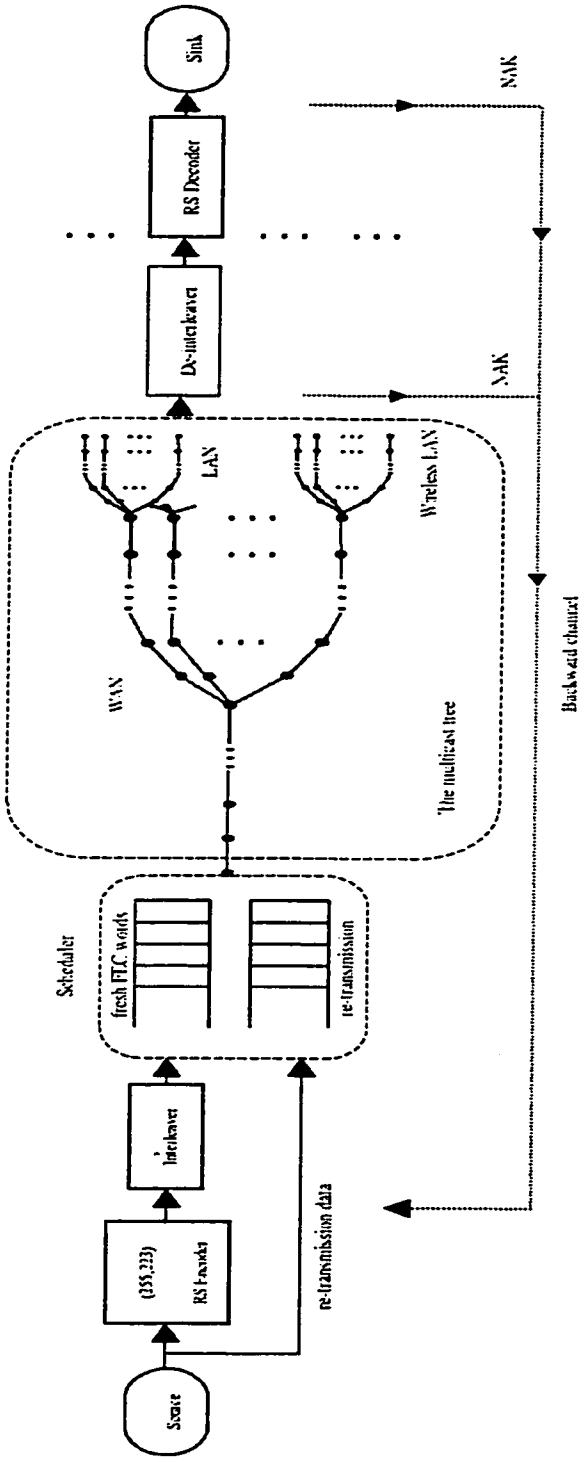


Figure 2-9: Selective Repeat ARQ

2.2.2.3 Hybrid error control (ARQ/FEC)

Studies show that integrating FEC and ARQ can improve multicast transmission efficiency and scalability [16,20]. A major difficulty when using FEC is to choose the right amount of redundancy in face of changing network conditions. Also, sending redundant data consumes additional bandwidth. In order to overcome this problem, ARQ and FEC can be used in combination. In many cases, proactive repair can reduce feedback implosion and the expected delay of reliable delivery without increasing overall bandwidth usage.

Figure 2-10: shows the hybrid FEC/ARQ system model.



2.2.3 Sender-Initiated vs. Receiver-Initiated Error Recovery

2.2.3.1 Sender-Initiated Error Recovery

Sender-initiated reliable multicast protocols require the sender to have a full knowledge of the receivers[7]. The sender then maintains a list, called the ACK list, for each transmitted packet. The ACK list is simply a list of the receivers from which ACKs have been received. Each time a receiver correctly receivers a packet, it returns an ACK to the sender. Upon the receipt of the ACK, the sender updates the ACK list for the corresponding packet. Lost packets are detected via the use of timers. When the sender transmits a packet, it starts a timer. The ACKs should then be received within a specific configurable time. If the timer goes off before an ACK is received from any receiver, then the sender assumes that the packet to that receiver was lost, and hence it retransmits the packet to the receiver and starts the timer again.

In most traditional protocols, when the sender wishes to transmit/retransmit a packet, it multicasts it to all receivers. When a receivers a packet correctly, it returns the ACK to the sender over a point-to-point connection. The sender uses a selective repeat approach for retransmissions; only lost/corrupted packets are retransmitted.

The sender-initiated approach could be the most appropriate in situations where the sender must be in entire control or must have a full knowledge of the receivers. However, the approach has a major problem: implosion. As the number of receivers increases, the number of ACKs received at the sender is also increased. After some threshold, the number of returned ACKs may exceed the ability of the sender to process them. Hence, the approach must be limited to small-multicast groups, unless some techniques are adopted to reduce the number of ACKs returned to the sender.

2.2.3.2Receiver-Initiated Error Recovery

With receiver-initiated error recovery, the responsibility of ensuring reliable packet delivery is placed on the receivers. The sender just needs to continue transmitting new packets until it receive a negative acknowledgment (NAK) from a receiver. Each receiver is responsible for maintaining a reception state for itself. Upon the detection of an error, the receiver sends a NAK to the sender indicating the error, Typically, a receiver detects a packet loss via the use of sequence numbers. If the receiver receivers a sequence number that is large than expected, it presumes the occurrence of an error and hence sends a NAK. The sender then retransmits the requested packet to that receiver via the unicast connection between both of them, or possibly via multicasting the packet to a group of receivers that includes the requester.

In order to secure against the loss of the NAK, the receiver uses timers in a fashion similar to the one used by the sender in the sender –initiated approach. In special cases where the sender does not have packets to send, it may be necessary for the sender to multicast periodic state information, giving the sequence number of last transmitted packet for example.

2.2.3.3 Hybrid Scheme

To overcome these limitations, a hybrid scheme can be designed by adding to the sender initiated approach characteristics of the receiver-initiated one. There are two major modifications required the first one is to include negative acknowledgments, allowing receivers to detect losses and report such losses to the sender. This may reduce latency significantly, since potentially the sender is able to detect a loss and retransmit a packet much sooner than by timeout alone. The second change regards scalability: to avoid the feedback implosion, the volume of feedback packets returned by receivers (now both ACKs and NAKs) must be reduced, and preferably their

arrival should be spread in time. The resulting hybrid schemes differ in how frequently feedback packets are returned by receivers, and how much status each feedback packet contains. The next section describes several of such hybrid feedback schemes, which are designed to reduce the amount of feedback packets and reduce the risk of implosion losses

2.2.4 Implosion avoidance

Different strategies have been used in order to overcome the implosion problem. They can be roughly divided into four categories (which can be used together)[10]:

- Tree-based
- Period-based
- Delay-based
- Polling-based

In the tree-based schemes□ receivers are organized according to a tree structure. The amount of feedback sent to the source is reduced because receivers send feedback to their parent node only. The actual rate of feedback packets arriving at a parent node depends on three factors:

(a) The degree of the node in the tree, i.e., how many receivers are sending feedback to the parent. (b) The rate in which data arrives at these (child) receivers; and

(c) The kind of protocol used in the interaction between sender and receivers (e.g., sender or receiver initiated, etc)

In the period-based scheme receivers send fewer feedback packets, but each feedback has more content (and is thus larger). The scheme appears in two forms: block acknowledgment or periodic acknowledgment. In the former case, the sender divides the transmission in blocks, and at the end of each block receivers send a "block ACK" containing several positive and negative ACKs referring to all packets in the block. In the latter case, each receiver periodically sends (with

fixed time period) a feedback packet containing a bit vector identifying which packets require retransmission. A receiver generates a feedback packet to its parent every "Tack".

The use of probabilistic timers is the main characteristic of the delay-based schemes. Feedback packets are multicast to the group, and so are the resulting retransmissions. When a receiver is given a NAK for a data packet it possesses, it may multicast a retransmission. When a given loss is experienced by more than one receiver, it is possible that multiple receivers multicast NAK packets. Delays are part of a feedback suppressing mechanism, which reduces the number of redundant NAK and retransmissions, preventing implosion (and congestion).

The last category is the polling-based implosion avoidance. It can be probabilistic or deterministic. One example of the former case is to avoid congestion and overrunning of receivers, the sender requests (through polling) feedback packets from a random subset of receivers, and uses such feedback in order to adjust its sending rate; asking all receivers to return such feedback would cause implosion. In the second case, the sender achieves reliable delivery and flow control with a sender-initiated ARQ scheme; the sender uses polling to select a subset of receivers to send an ACK packet, so that the volume of ACK packets is sender-controlled and does not cause implosion.

2.2.5 Organization

Most current reliable multicast protocols use one IP multicast group to propagate packets from the sender to receivers. Packets flow through a multicast routing tree, which is built by the network according to packet routing (e.g. the Distance Vector Multicast Routing Protocol DVMRP) mechanics and a membership protocol (e.g. Internet Group Management Protocol IGMP) [18].

Most protocols deliver packets to receivers using the above scheme, they however differ in how feedback is propagated from receivers to the sender: there are three general organizations (see Figure 2-11)

·Fat or centralized model

· Hierarchic or tree-based

· Symmetrically distributed

This organization permeates in the protocol design, is affecting error control, implosion control, flow Control, and congestion control.

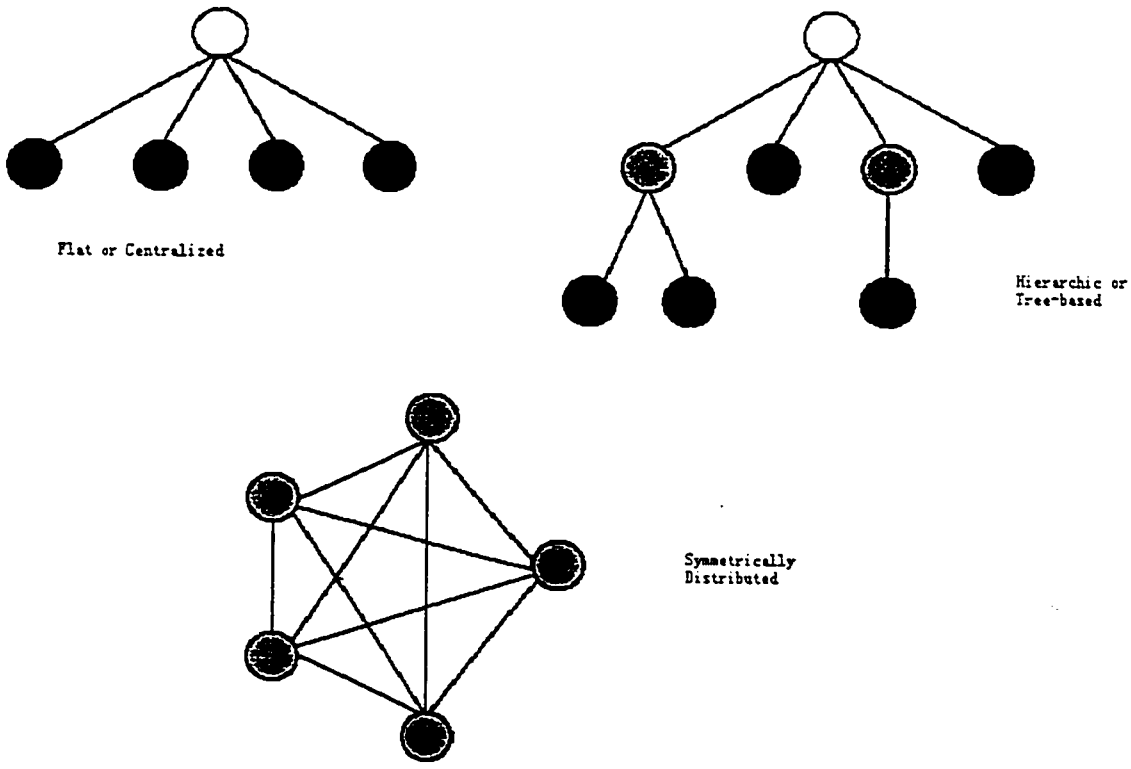


Figure 2-11: Origination of Reliable Multicast Protocols

In the simplest organization, the fat or centralized model, each receiver directly interacts with the sender. That is, receivers transmit feedback to the sender, which examines the feedback and carries out recovery by retransmission. One potential benefit of such a model is that the sender has information about all receivers and is able to make "global" decisions regarding the entire group. For example, when retransmissions are required, the sender is able to judiciously decide between multiple unicast and multicast retransmissions according to the number of receivers that experienced a given loss. With the loss scenarios of Section2.2.1 in mind, using global multicast or multiple unicasts to recover losses may have a strong impact on network cost associated with the protocol. On the other hand, this flat model presents scalability problems. As the sender handles feedback packets from all receivers, the volume of feedback packets may exceed the capacity of the host and its surrounding network, leading to implosion losses. Further, in wide-area networks, where the RTT (round trip time) [8] between sender and receiver may be substantial, a feedback packet and the resulting retransmission has to propagate all the way from receivers to the sender and then back; this is likely to increase end-to-end latency.

The other two models depend on the interaction among receivers. In the hierarchic or tree-based model, receivers are organized according to a tree, with the sender at the root. Every receiver is a child node, which sends feedback to its parent node. This increases the scalability of the protocol: since the sender only receives feedback packets from its child nodes, the problem of implosion is somewhat alleviated. In the tree-based model, there are two ways of organizing the recovering of packets:

· Collated feedback: with centralized retransmission, all acknowledgments are collated through the tree and the sender does all retransmissions according to the collated ACKs

· Hierarchical recovery: each parent node in the tree keeps packets received in buffers and performs retransmissions requested by its children; if a parent does not have a requested packet, it asks its own parent.

Though there are difficulties associated with both schemes (which will not be addressed here), hierarchical recovery seems to be more common. With collated feedback with centralized retransmission, packets have to be NAKed all the way to the source, and then retransmitted, an operation, which may involve substantial delays depending on the network distance between the two. The gain in comparison to the above at scheme is that feedback is collated through the tree, reducing the risk of implosion, and reducing the network cost (feedback packets go only to the parent). Hierarchic recovery allows localized recovery to take place: a given loss may be recovered by a nearby node (often called "representative" or "group leader"), with low latency and without affecting the rest of the group.

In the symmetrically distributed model, any receiver, which has received the packet (or the sender), is capable of retransmitting it. All retransmission requests are multicast to the entire group, so that all receivers learn about it. One or more receivers which received the packet will retransmit, again via global multicast. This symmetrical model has high network cost: suppose one of the N receivers experiences a loss, and multicasts a NAK packet; as a result, the N –1 node that received the packet retransmit via multicast. This corresponds to (N-1)*N packets being exchanged, a very high number for a single loss.

To overcome the network cost problem, a "feedback and recovery suppression mechanism" can be used. Random timer values are used to delay receivers to attempt to reduce the amount of redundant NAKs and retransmission packets. To calculate a good random suppression timer for a given loss, each receiver needs to estimate the distance between itself and the sender of the packet.

If all participants can send, each of the receivers needs to keep RTT estimates between itself and all others. To estimate RTT between itself and all other receivers, each receiver needs to exchange periodic messages with all other receivers. The symmetric model better suits many-to-many applications, where the overhead may be reduced because group members (frequently) exchange (data) messages.

CHAPTER 3

Effects of DR location on QoS Performance of Reliable Multicast

3.1 Objective and Motivation

As, we discussed in the last chapter, the main reason for the packets loss, delay and jitter encountered in the Internet is due to congested routers. Applications, especially real-time multicast, should take good care of this feature of the Internet to achieve a good performance. A new Hybrid FEC/ARQ Reliable Multicast Protocol was introduced.

Choosing the proper Domain Receiver position and buffer size is a challenge for a network designer. The domain receiver position is directly related to the traffic on each router, link, retransmission time, and as a result and affects throughput and delay to each destination. Buffer size is another important parameter, for it is directly related to the over flow in each buffer, which can affect decrease, the whole network performance. Network traffic is also an important parameter in network design; suitable traffic will lead to a good performance of the network. In our simulation, we will try different input parameters to find better out put parameters in the Hybrid FEC/ARQ source tree reliable multicast technology

3.2 Description of Hybrid FEC/ARQ source tree reliable multicast technology

We introduce a new technique that integrates word interleaving, Forward Error Correction, and Automatic Repeat Request to mitigate the error and loss effects encountered in wire and wireless Internet application. The most prominent of these is multimedia multicasting where stringent requirements on delay and delay jitter are to be met. For multicast video sessions spanning tens of routers, the integration of interleaving, FEC and ARQ, as well as fine-tuning of the various parameter of each mechanism, will play a major rule towards success and further development of Qos routers. Proceeding requirements on QoS who have to handle not only the regular IP unicast traffic, but also multicast routing, group management, bandwidth reservation, and the integrated error concealment measures above.

3.3 Simulation Procedure of Source Tree Multicast

3.3.1 Assumption

In order to focus on the effects of DR location on QoS Performance of hybrid FEC/ARQ, we make several simplifying assumptions about the network model. We assume that link loss rates are not affected by the rate at which the sender transmits. This is reasonable in a scenario where the congested links utilized by the protocol are also utilized by many other sessions. We consider the case that all the receivers have the same loss rate. NAKs are sent using UDP ports only. These assumptions make the system model simpler to evaluate.

3.3.2 Input and Output Parameters

3.3.2.1 Input Parameters

There are two kinds of input parameters, one is fixed constants, and the others are variable input. The fixed constants are unchangeable within the whole simulation; however, for the variables inputs, by assigning them different values, we can achieve different performances.

The fixed constants are:

Sender number: there are five senders in our simulation, which involves a hypothetical network.

Router number: there are ten routers in our simulation, which involves a hypothetical network.

Destination number: there are seventeen receivers in our simulation, which involves a hypothetical network.

Pc: this is defined as the probability of packet successful reception in the buffer. We fix it as 99%

ρ: this is defined as the traffic rates of sender. In this simulation, ρ is set to 1, means stream mode.

The input variables are:

Max_BufferSize: each router maintains a buffer that can store Max_BufferSize.

TTL: when a node generates a data message, it will be stored into the temporary buffer and awaits transmission. After a certain time, if the message hasn't reached its destination before its TTL, then a certain indication is set. This is an input parameter. Every packet has a living time. If the time expires, it will not reach the receiver and the network will drop it automatically. In my simulation, I set it as 62ms.

Generation rate: Senders send next packet after a certain interval, this makes for different generation rate for each sender, and different source traffic.

3.3.2.2 Output Parameters (Performance Criteria)

There are many parameters to be considered. Changing any one of these will affect the final performance. The following performance criteria are defined in order to evaluate the performance. Each performance criteria are averaged over all the program iterations and over the number of nodes.

3.3.2.2.1Average and Variance of Total End to End Transfer Delay

Total Transfer End to End Delay is defined as the total time that one packet takes from its generation to its successful delivery to the final destination. This includes the time it waits in the various buffers (Queuing Delay) from the source to the destination as well as the various Propagation Delays and times for retransmission. The Mean of Total Transfer delay is calculated by dividing the sum of all total transfer delays of all successful messages of all nodes by the number of these successful transmitted messages. The Variance of Total Transfer Delay is calculated by dividing the sum of the squares of the differences between the Average Transfer Delay and individual Transfer Delay by the total number of successfully transmitted messages. The formula of Average total Transfer Delay (ATTD) and Variance of Total Transfer Delay (VTTD) are listed as follows:

$$ATTD = \frac{\sum_{i=1}^{n} TTD_i}{n}$$

$$VTTD = \frac{\sum_{i=1}^{n} (ATTD - TTD_i)^2}{n-1}$$

Where n is total number of successfully transmitted messages to all receivers during the whole simulation program, TTD represents the Transfer Delay of each successfully transmitted message. In our simulation, this equals the difference between final delivery iteration number and generation iteration number. These two numbers represent the generation time and final arrival time of each message respectively.

3.3.2.2.2Average and Variance of buffer Overflow

Each flow in the subject router has a buffer, once a message is generated or received from his neighboring router; the message is inserted into the buffer and queued for transmission. If the number of queued messages exceeds certain limit, then there is buffer overflow. We set a counter to record the number of instances of buffer overflows. The mean of the buffer overflow is calculated by dividing the sum of these overflows by the total simulation iterations and by the number of routers. Variance of the buffer overflow is calculated by dividing the sum of all squares of the differences between the average buffer overflow and each router buffer overflow by the total simulation time (number of iterations)

and by the number of routers. The following formulas are for Average Buffer Overflow (ABO) and Variance of Buffer Overflow (VBO).

$$ABO = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} BO_{ij}}{mn}$$

$$VBO = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} (ABO - BO_{ij})^2}{(m-1)(n-1)}$$

BO is the buffer overflow number in i^{th} iteration and j^{th} nodes, (equals 1if buffer overflow takes place and "0" if there is no buffer overflow) m is the number of iteration and n is number of nodes.

3.3.2.2.3Average End-to-End Throughput

The average End-to-End throughout is defined as the ratio of the number of packets that are successfully transmitted in a very long interval to the maximum number of packets that could have been transmitted with continued transmission on the channel. By the simulation model, we write the average throughput as:

$$AT = \frac{total\ \ number \cdot of\ \ successful\ \ data\ \ messages}{total\ \ number\ \ of\ \ generated\ \ data\ \ message}$$

3.3.3 Simulation Data Structure

Packet data structure:

In the simulation, each message contains seven fields

1) Sender identity (Multicast Group Number): There are five senders in our simulation.

2) Receiver identity: There are seventeen receivers in our simulation.

3) Multicast Signal: This field relates to the packet whether it is Multicast or not. It will decrement by 0 mean multicasts to all the receivers of this flow, when the value reaches 1, the packet only sends to one receiver as the receiver field indicates.

4) Sequence Number: Each packet has a unique Sequence Number (0~1000000) to indicate the sequence. Simulation time is 1000000 packets times and is divided into many windows, each windows has 256 packets. During each window we average the results of all performance results.

5) Generation Time: Each packet has a generation time when it is generated, if any router loses the packet, and needs retransmission; the retransmission router (Domain Receiver) will give a new generation time.

6) Accumulated time: Accumulation time represents the total transfer delay of each successfully transmitted message. In our simulation, this equals the difference between the final delivery iteration number and the generation iteration number. These two numbers represent the generation time and final arrival time of each message respectively.

7) TTL (Time to live): Time to live indicates the longest time the packet could survive in the simulation. If TTL exceeds this period of time, it will be dropped.

NAK Data Structure:

1) Sender identity (Group number): There are five senders in our simulation.

2) Receiver identity: There are seventeen receivers in our simulation.

3) Multicast Signal: This field determines if the packet is Multicast or not. It will decrement by 0 mean multicasts to all the receivers of this flow; when the value reaches 1, the packet only sends to one receiver as the receiver field indicated; when the value reaches 2, the packet will be a NAK packet.

4) Sequence Number: Each packet has a unique Sequence Number (0~1000000) to indicate the sequence. Simulation time is 1000000 packets times and is divided into many windows, each windows has 256 packets. During each window we average the results of all performance results.

5) Mean Accumulation time: Mean Accumulation time represents the mean of the transfer delay of each lost packet needed for retransmission in this NAK packet. In our simulation, this equals the difference between final delivery iteration number and generation iteration number. These two numbers represent the generation time and final arrival time of each message respectively.

8) TTL (Time to live): Time to live indicates the longest time the packet could survive in the simulation. If TTL exceeds this period of time, it will be dropped.

Multicast Router table:

In reference to multicast function of the hypothesize Internet clusters of Figure 3.1, the following shows the shortest path routers from each of the senders to all 17 receivers. For example, "R1 S1 R4 DG" means packets from flow of S1 passes by router R1 and next hop is R4 and final destination is specific router DG.

R1 S1 R4 DG //Current router source next hop destination

R1 S1 R4 DD

R1 S1 R4 D6

R1 S2 D1 D1

R2 S2 R1 D1

R2 S2 D2 D2

R2 S4 R4 D7

R3 S2 R2 D1

R3 S2 R7 DE

R3 S2 R2 D2

R3 S2 R7 DA

R3 S3 D8 D8

R4 S1 R5 DG

R4 S1 R6 DD

R4 S1 D6 D6

R4 S4 R2 D7

R4 S5 D3 D3

R5 S1 DG DG

R5 S3 R9 D8

R5 S3 R9 DB

R5 S3 R8 DF

R5 S4 R4 D7

R5 S4 R4 D4

R5 S4 R9 DI

R5 S5 DH DH

R6 S1 DA DA

R6 S5 R4 D3

R6 S5 D5 D5

R6 S5 R7 D9

R7 S2 R9 DE

R7 S3 R3 D8

R7 S4 DI DI

R7 S5 D9 D9

R8 S3 DF DF

R8 S4 R5 DI

R8 S4 R5 D7

R8 S4 R5 D4

R9 S2 DE DE

R9 S3 RA DB

R9 S3 R7 D8

R9 S4 R7 DI

R9 S5 R5 DG

R9 S5 R6 D5

R9 S5 R6 D3

R9 S5 R6 D5

R9 S5 R6 D9

RA S1 DD DD

RA S3 DB DB

Unicast Routing table:

Unicast router table is also in reference to unicast function of the hypothesize Internet clusters of Figure 3.1, shows the shortest path routers from each of the senders to all 17 receivers. For example, "S1 R1 R4 R6 RA DE" means packets from flow of S1 passes by router R1 and next hop is R4, R6, RA, and final destination is specific router DE.

S1 R1 R4 R6 RA DE

S1 R1 R4 R5 DG

S1 R1 R4 D6

S2 R3 R2 D2

S2 R3 R2 R1 D1

S2 R3 R7 R9 DE

S2 R3 R7 DA

S3 R5 R8 DF

S3 R5 R9 RA DB

S3 R9 R7 R3 D8

S4 R8 R4 D4

S4 R8 R5 R9 R7 DI

S4 R8 R5 R9 R7 R3 D7

S5 R9 R5 DH

S5 R9 R6 D5

S5 R9 R7 D9

S5 R9 R6 R4 D3

Buffer Data Structure:

1) Source Name: Source name used to indicate which sender this buffer belongs to.

2) Buffer Size: Buffer size is used to indicate how big the buffer is.

3) Lost Packet Counter: It is used to indicate how many packets are lost in this buffer.

4) Total Packet Counter: It is used to indicate how many packets enter into this buffer.

Temporary Buffer Data Structure:

Temporary buffer for the first packets in the buffer

1) Source Name: Source name is used to indicate which source this buffer belongs to.

Destination Single Window Data Structure

1) Total Accumulation time: Total Accumulation time is used to indicate total accumulation time for all packets in a single window.

2) First Packet Sequence Time: it is used to indicate first enter packet sequence time to this window

3) Last Packet Sequence Time: It is used to indicate the most recent sequence time packet in this window.

4) Maximum Packet Sequence Time: It is used to indicate the Maximum sequence time packet in this window.

5) Total Packet Count: It is used to indicate total packets received in certain time window.

Loss Packet Buffer Data Structure:

1) Source ID: Source name is used to indicate which source this buffer belongs to.

2) Sequence Time: It is used to indicate which packet needs retransmission.

3) Delay Counter: It is used for each buffer to provide a delay time, which allows the same sequence time packet to only be retransmitted once.

3.3.4Source Tree Multicast Simulation Model

The source tree multicast is different from the shared tree multicast. In source multicast, all the retransmission is done by source; and in the share tree multicast, there is only one rendezvous point in the whole network; this rendezvous point does all the retransmission. For the source tree multicast, the rendezvous point on its source tree does the retransmission, and the rendezvous is also called domain receiver. The Domain Receiver is the core of a source based tree. It has four functions: to store packets from source, to multicast or unicast packets to designed receivers, to collection of the Nak for receivers, and to make a decision to multicast or unicast the retransmission.

The simulation network structure is shown on Figure 3.1. The hypothetical network has five sends, ten routers, and seventeen receivers.

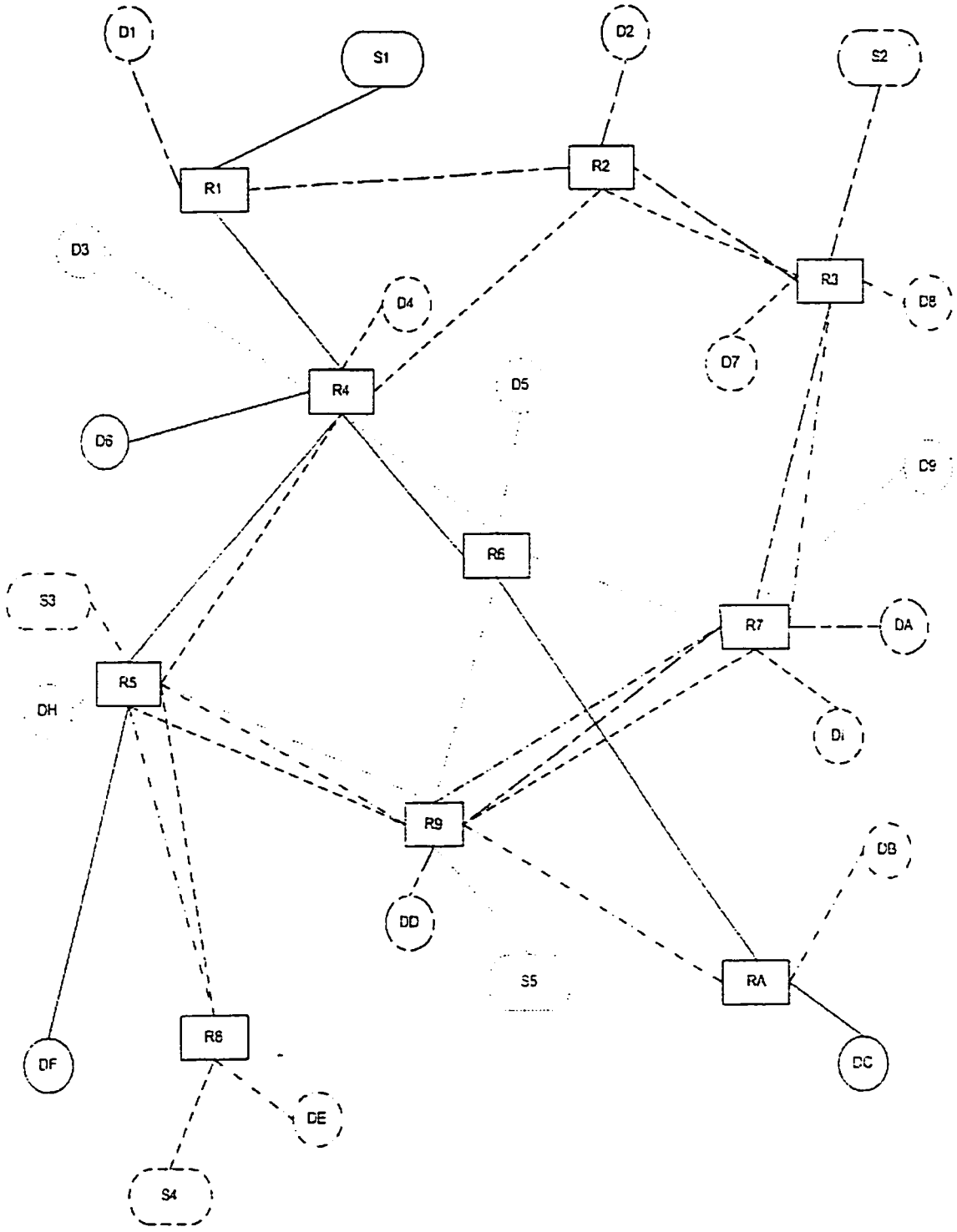


Figure 3-1: Simulation Network Bed Structure

We used UML (Unified Modeling Language is the industry-standard modeling notation for object-oriented systems, and is the premiere platform for rapid application development) object original design where each network entity to include a sender, some routers, and some destinations.

The source in our simulation acts a creator of each packet.

The router includes the dispatcher, the receiver, the sender, the bufferque, and the retransmiter.

(1) The router receiver gets a packet from the UDP port.

(2) The router bufferque will check the source in the receiver packet.

(3) The router dispatcher is responsible for queuing the packets on the router waiting buffer.

(4) The router retransmitter used to check every packet in the NAK queue and decide to Unicast or Multicast.

(5) The router sender used to send the request packet to the UDP port

The destination includes the sender, the receiver the recorder and the winmanager.

(1) The destination receiver is used to get a packet from the UDP port.

(2) The destination adaptor is used to check new packet sequence number to adjust create a new receive window or call operator in current receiver windows.

(3) The destination windows end is used to set free receive windows.

(4) The destination windows manager has two major functions: built loss queue; decided which NAK packet to send.

(5) The destination sender used to send the request packet to the UDP port

(6) The destination recorder is used to create data file from simulation.

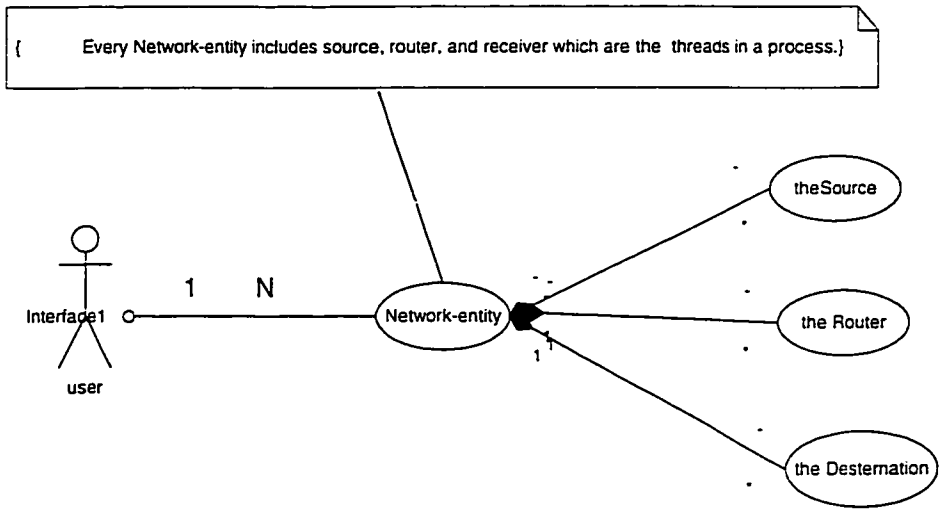


Figure 3-2: Simulation User Case

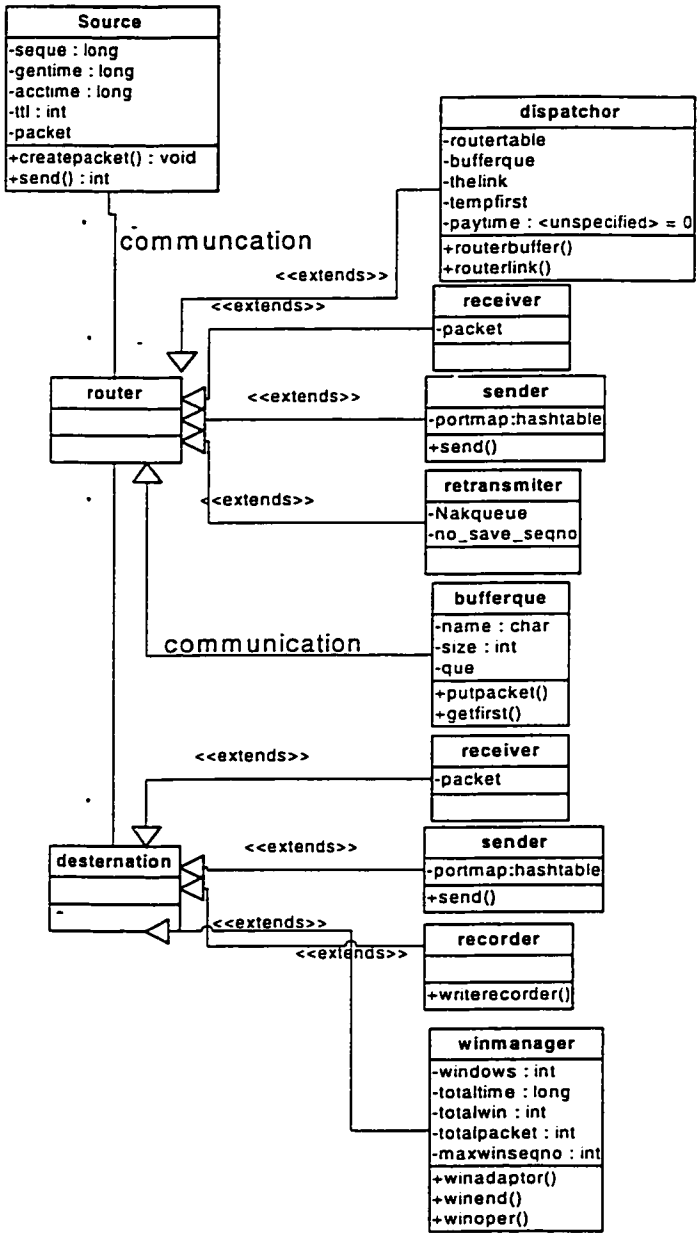


Figure 3-3: Class Description

3.3.4.1 Source Function

The source in our simulation acts a creator of each packet. We initialize the value of the sequence number "sequno" to 0 and generation time "gentime" to 0, accumulation time (acctime) to 0, time to live "TTL" to 62 for the first generated packet. In each send loop, we increase the "seqno" by one, but initialize the acctime is 0. Each packet has different generation time, which is equal to current program iteration. Then, we assign the seqno, gentime, accutime and TTL to this packet, following by a call to send function to send it. To simulate the real situation of multicast network, we have to generate a large number of packets in our simulation. The loop counter is set as 1e6; it means we generate 1000000 packets in our simulation. If the counter is bigger than 1e6, the source will stop working, as in figure 3-4.

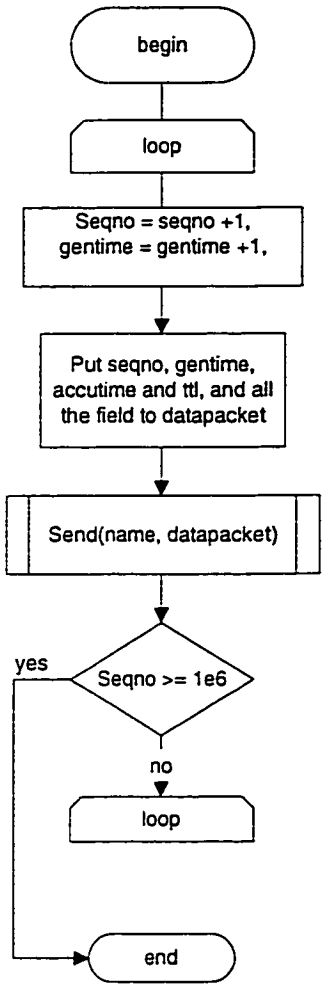


Figure 3-4: Source Function

3.3.4.2 Router-receiver Function:

When the router receiver gets a packet from the UDP port, it will check whether the packet is a data packet or a NAK packet. If the received packet is data, it will call the router bufferque, and put the data into the data buffer. If the received packet is a NAK packet, it will put it in the retransmissions NAK queue, as in Figure 3-5.

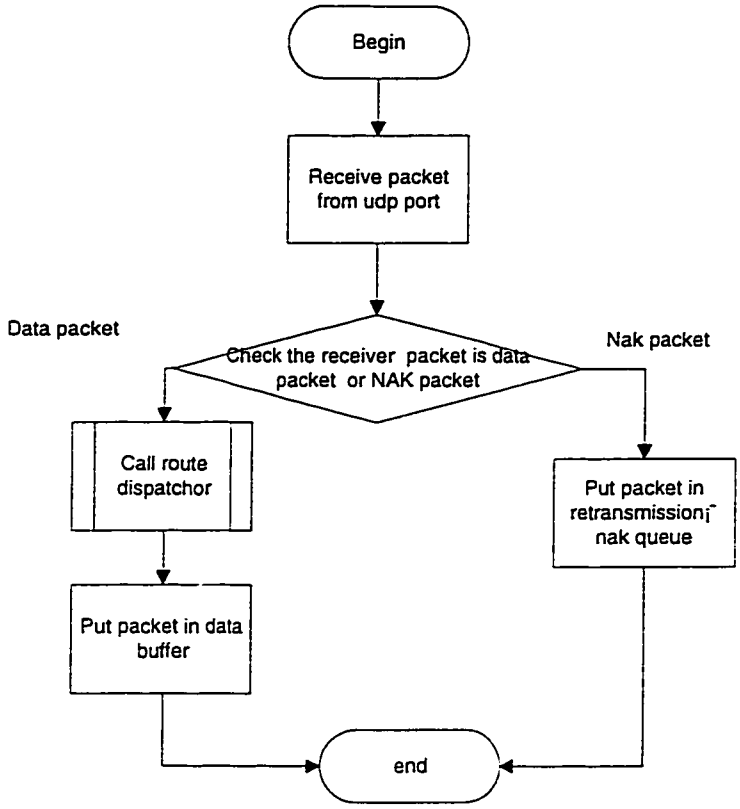


Figure 3-5: Router-receiver Functions

3.3.4.3 Router-bufferque Function

The router-bufferque will check the source in the receiver packet. There are five source flows in our simulation. The packet will be sent to a certain buffer matched with the packet source. The packet can be lost in two cases: one is background noise; another is the buffer is full condition. The background noise follows the uniform distribution. One

percent of all packets will be lost in the worst-case noise environment. But the surviving packet also can be lost if the buffer is full. In this simulation, we use two counters to compute numbers of correctly received packets and lost packets. For the surviving packet, we increase the accumulation time by one iteration plus 1. Finally, we put the packet in the buffer end, as in Figure 3-6.

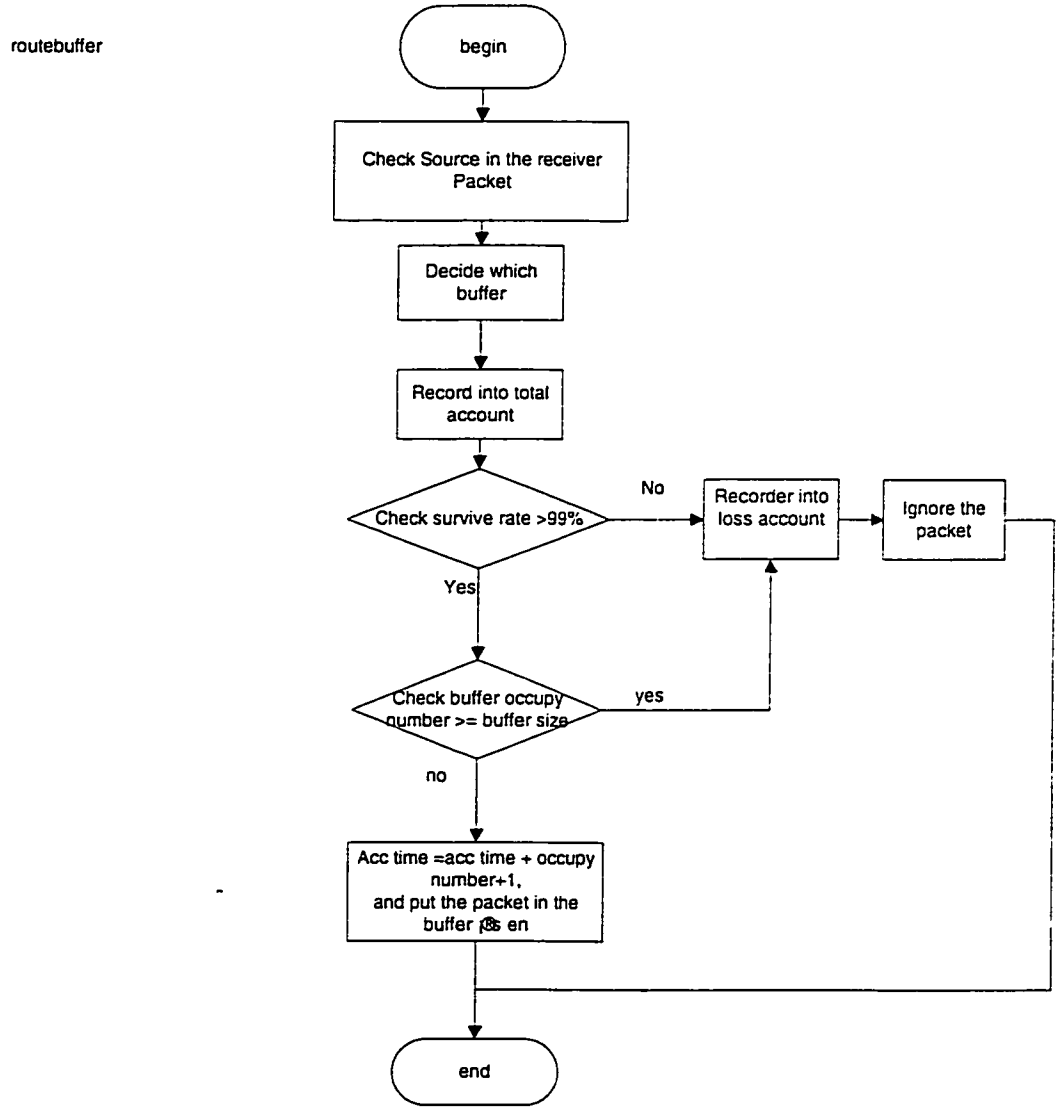
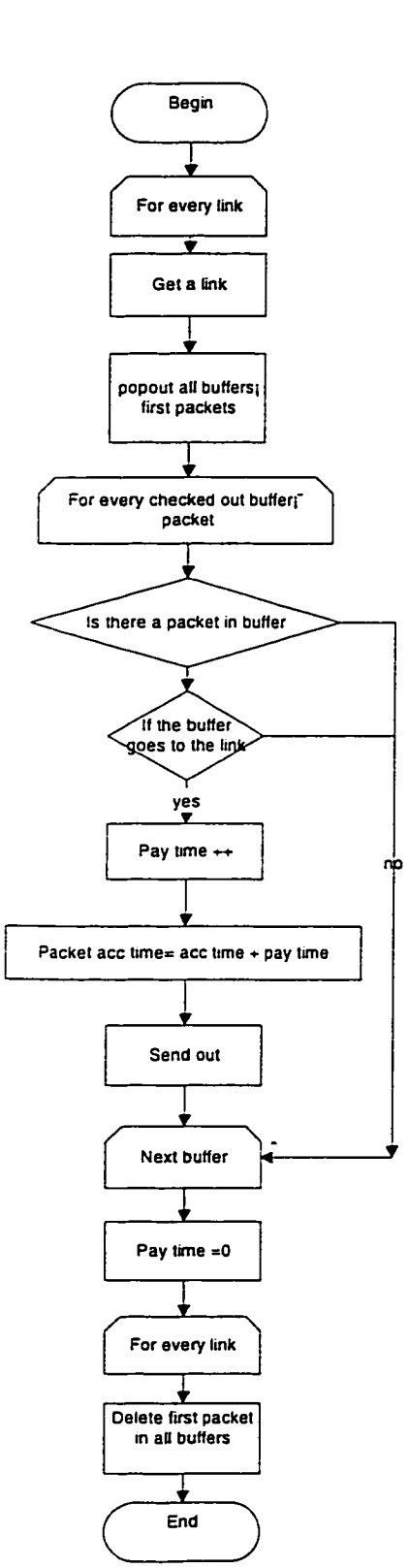


Figure 3-6: Router-bufferque Function

3.3.4.4 Router-dispatcher Function

Assuming all links have the same capacity, and all packets of different flows from this router to the next router share this link, then all of the buffers, which want to share the link, can pop up its first packet of queue into the waiting buffer. For every checked out buffer's packet that wants to use this link, its packet accumulation time will be increased by the link service time (pay time). Then it sends out this packet to the next router receiver. In each loop the waiting buffer and pay time will be set free after the loop finishes. In each round of service from flow buffers to waiting buffer, one packet from buffer is sent to waiting buffer one after another. The pay time delay component accounts for such sequenced transition and is added to acctime of each packet, as in figure 3-7

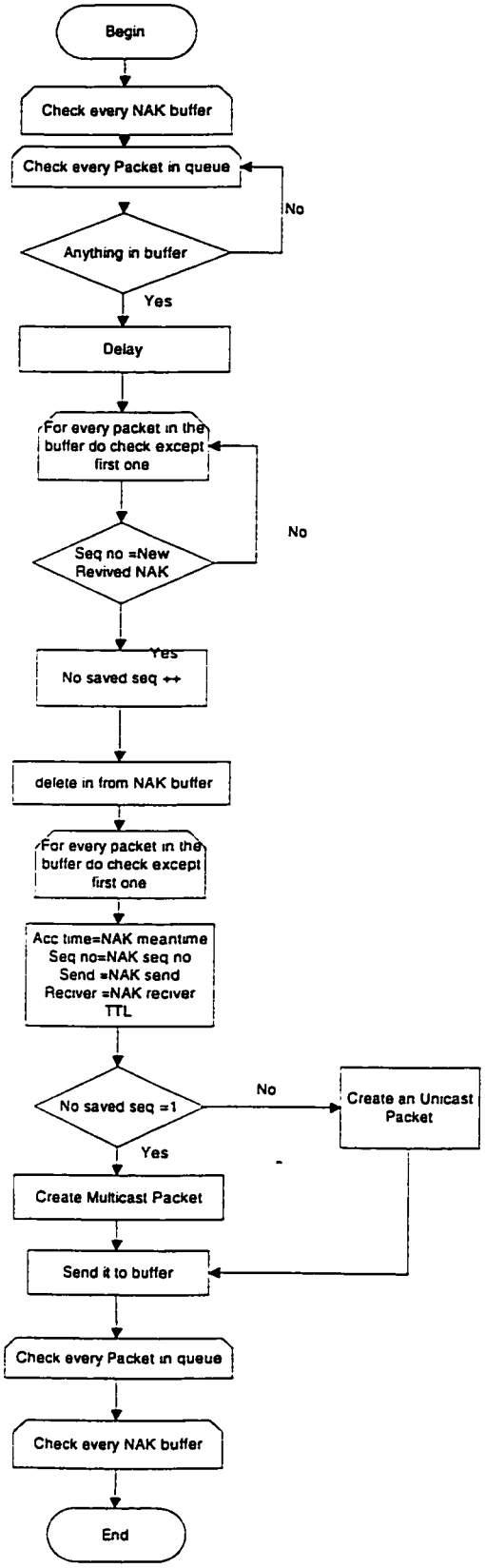
Figure 3-7: Router-dispatcher Function



3.3.4.5 Router-retransmitter Function

Router has a NAK buffer queue for each flow into it. The router retransmiter will check every packet in the queue. If it finds some NAK packet in the queue requesting retransmission of some packet, it will record it and allows 10ms delay of waiting in case there is another request for the same packet. If it receives another NAK packet request for same packet, it will be recorded, and number of NAK in simulation will increase by one. The resent packet ACC time will be initially set to worst NAK generation time by worst we mean over all received who have generate a NAK to this packet , Sequence number will be NAK sequence number, Sender will be NAK sender, Receiver will be NAK receiver and TTL is set to 62. Then retransmitter will check the number of NAKs already received to this packet; if it is equal to one, we send a Unicast packet; if it is larger than one, we send Multicast packets, as in Figure 3-8.

Figure 3-8: Router-retransmitter Function



3.3.4.6 Router-sender Function

When the buffer needs to send a packet, it first checks the receiver ID field of this packet. Then it checks the Hash table for the UDP port, and then sends the packet to the UDP port, as in Figure 3-9.

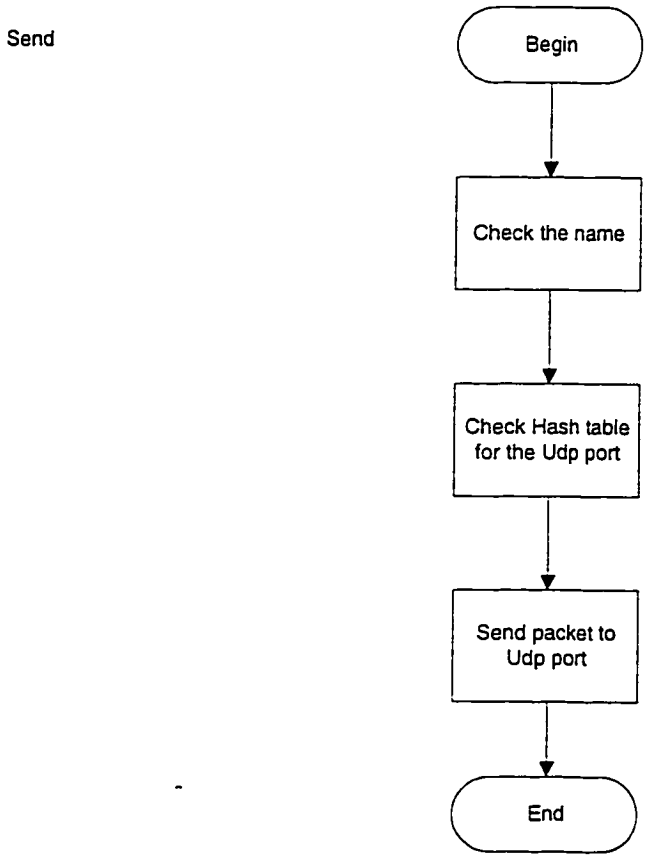


Figure 3-9: Router-sender Function

3.3.4.7 Destination- receiver Function

The Function of the destination receiver is very simple. First it gets the packet from the UDP port, and then delivers it to the windows manager, as in Figure 3-10.

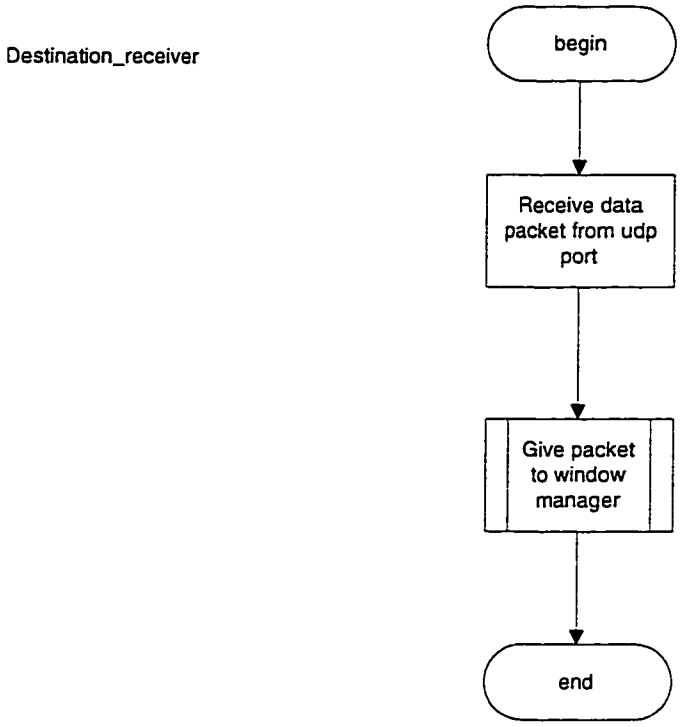
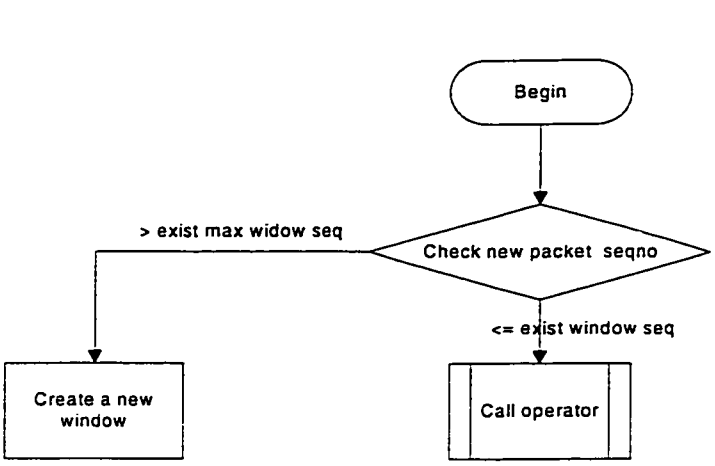


Figure 3-10: Destination- receiver Function

3.3.4.8 Destination- adaptor Function

The adaptor will check the new packet sequence time; if it is larger than the existing maximum sequence time of the current window, we create a new window; if it less or equal to the maximum sequence time of the existing window, it will be called Operator, as in Figure 3-11.

Figure 3-11: Destination- adaptor Function



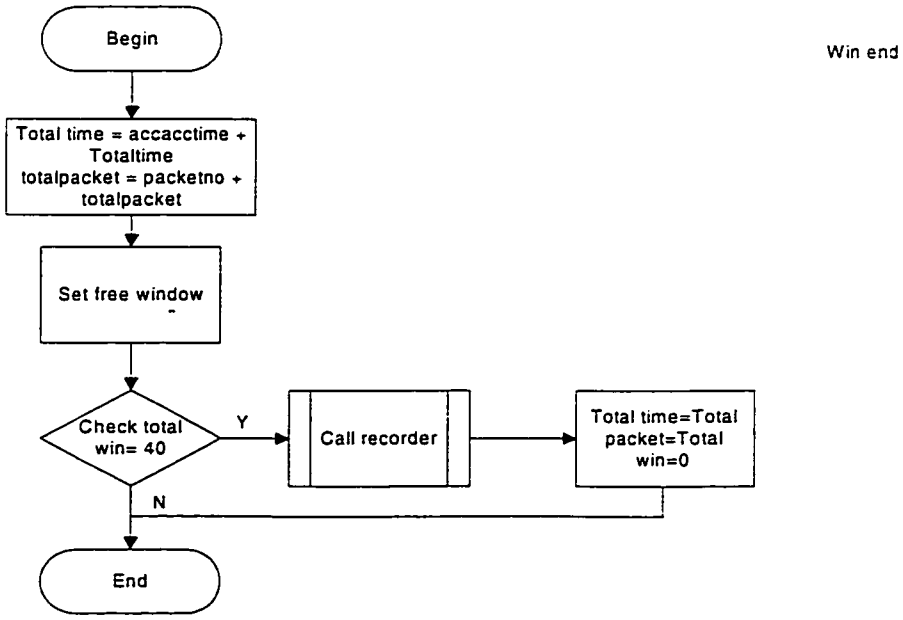


Figure 3-12: Destination-windows-end Function

3.3.4.9 Destination-windows-end Function

We define the total packets accumulated times at end of window as the current total accumulate times of previse packets times of last iteration, plus the accumulation time; and the total correct packet equal to the current total packet (of last iteration) plus the number of correct packets that reached the receiver, and then set free the window to start to compute the same performance criteria in the next window. When the total number of windows equals 40, we call the recorder and average the results of all windows, as in Figure 3-12.

3.3.4.10 Destination-windows-manager Function

The destination windows manager is the most complex in our simulation, including two functions.

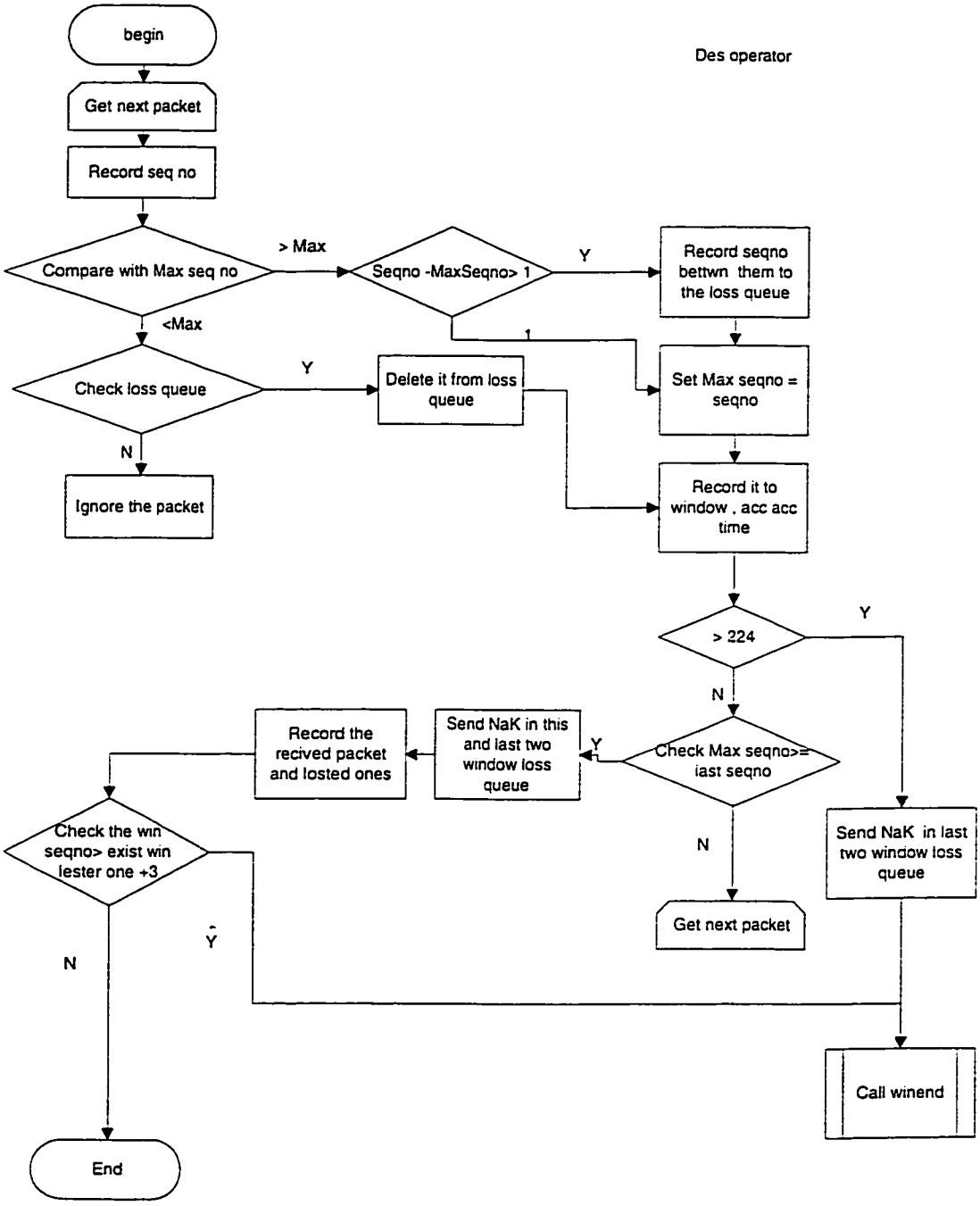
1) Building loss queue: It first gets the packet from the destination receiver, then records the sequence number of this packet, and compares it with the Maximum sequence number of the current window. If it is larger than the maximum sequence number, then we subtract it from the window maximum sequence number. If this new number is larger than one, we record the sequence number in the loss queue, and then we set the maximum sequence number of windows as sequence number of this current received packet. If it is equal to one, we directly set the maximum sequence number as the sequence number of this packet, but we do not record any sequence number in the loss queue, means no retransmission is necessary for this packets .

If the sequence number of this packet is less than the maximum sequence of this window, we will check already exists in the loss queue. If there is no same sequence number packet in the loss queue, the packet is to be ignored, means we have already received the packet. If there is the same sequence number packet in the loss queue, delete it from the loss queue, means no retransmission is necessary for this packet, and record its accumulation time to the total accumulation time of this window.

2) Build NAK packet: The next item is that the window operator will check the packet sequence number in the current program window to determine if it is larger than 224 or not. If it is less than 224, we compare the maximum sequence number of record packet with the sequence number of the last packet in the current windows (last sequence number). If equal to or large than the last sequence number, send NAK to this and to the last two windows loss queue, and record the received packet number and the lost one. Then check the window sequence to determine if it is larger than existing third window.

If it is larger than three, then close first one, and call windows-end function. If the maximum sequence number is not larger than the last sequence number, keep receiving the new packets. There still is a special case if the packets number in the window is larger than 224; we will send NAK in the last two loss queues, and call windows-end to close this window, as in Figure 3-13.

Figure 3-13: Destination-windows-Manager Function



3.3.4.11 Destination-Sender Function:

When the buffer needs to send a packet, it first checks the receiver ID field of this packet, then checks the Hash table for the UDP port, and then sends the packet to the UDP port, as in Figure 3-14.

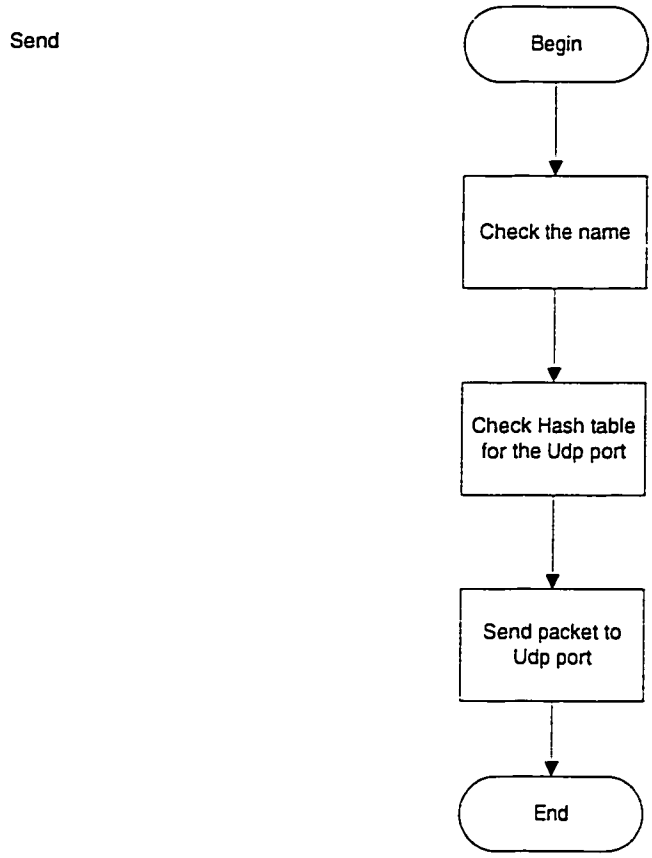


Figure 3-14: Destination-Sender Function

3.3.4.12 Destination-Recorder Function:

The Destination Recorder first gets the data from the window manager, then creates a file named with the name of the destination and source, and records the data on the file, as in Figure 3-15.

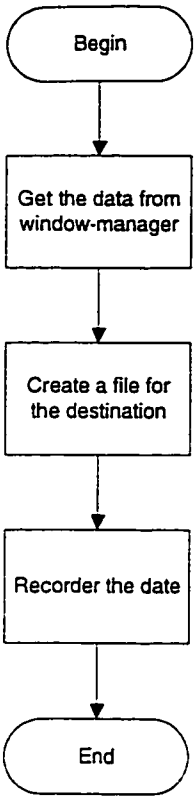


Figure 3-15: Destination-Recorder Function

3.4Simulation Results

There are a lot of input variables, and by their proper setting we can get a better performance in our simulation. Following are important input variables:

1) Buffer size

2) Domain Receiver (DR or Rendezvous Point) position

3) Input Traffic Intensity

We ran the simulation model programs with different values of the input parameters. In the following we compare the performances of the FEC/ARQ source tree Multicast with different input parameters.

3.4.1 Average End-to-End Total Transfer Delay (ATTD)

As we mentioned above, we define ATTD as the total time that one packet takes from its generation to its successful delivery to the final destination. This includes the time waiting in the various buffers (Queuing Delay) from the source to the destination, as well as the lost packet delays in the loss queue and the time it takes for retransmissions.

As shown in Figure3-16, we fix the buffer size and successfully receive probability (Pc) and the DR position and vary the source stream packets' generation rates; we can see the change of the average total transfer delay (ATTD) parameter. Simulations are run by varying the generation rates from 100, 125, 166, 250, to 500 packets per second and fixed buffer size =50 packets, Timeout =62, Pc=0.99 and the maximum number of iteration =1e6. It is seen that when the generation rates are low, the average total transfer delay is also low; when the input traffic becomes heavier; the average total transfer delay also increases, and the variance also changes as generation rates increase, as in Figure3-17. Moreover, each flow has different hops in our simulation. The fourth flow has the most hops among all flows, so flow four has worse performance.

Similarly, keeping the generation rates at 1000 packets per second, we can run the simulation as buffer size varies. If the buffer size is less than 30, some destination transfer delays are infinitely long for these large numbers of packets lost in the buffers, and cannot reach the destination. The average end to end total transfer delay will lead to infinity; we can not get the required data. We run the simulation at buffer sizes from 30, 35, 40, 45, 50, and fix other parameters. It is seen that when the buffer size is small, the average total transfer delay is low; when the buffer size increases, the average total delay also increases, but the variance decreases as buffer sizes increase, as in Figure3-18, 3-19. The fourth flow has the most hops

and passed 15 routers in our simulation. In fact, the fourth flow has 18 hop-to-hop links that are shared with other flows. So when the buffer size increases, the fourth flow has more variation in delay performance. As a result, variances increase as buffer size increases.

Next, we change the Domain Receiver position.

If we only change the Domain Receiver for first source, we can see if the DR is located at R1, we get less transfer delay than if located at R4 for the whole network; for variance point of view DR located at R4, is better than DR located at R1, as in Figure3-20, 3-21.

If we only change the Domain Receiver for source two, we can see that if DR is located at R2 or R7 yields less average transfer delay than if located at R3 for whole network, For variance point of view DR located at R3 yields better results than DR located at R2 or R7, as in figure, as in Figure3-22, 3-23.

If we only change the Domain Receiver for source three, we can see that if DR is located at R8 or R9 yields less average transfer delay than if located at R5 for whole network. For variance point of view DR locates at R5 is better than R8, R9, as in Figure3-24, 3-25.

If we only change the Domain Receiver for source four, we can see that if DR is located at R8 yields less average transfer delay than if located at R5 for the whole network. For variance point of view DR locates at R5 is better than R8, as in Figure3-26, 3-27.

If we only change the Domain Receiver for source five, we can see that if DR is located at R6, R5 yields less average transfer delay than if located at R9 for whole network. For variance point of view DR locates at R9 is better than R6, R5, as in Figure3-28, 3-29.

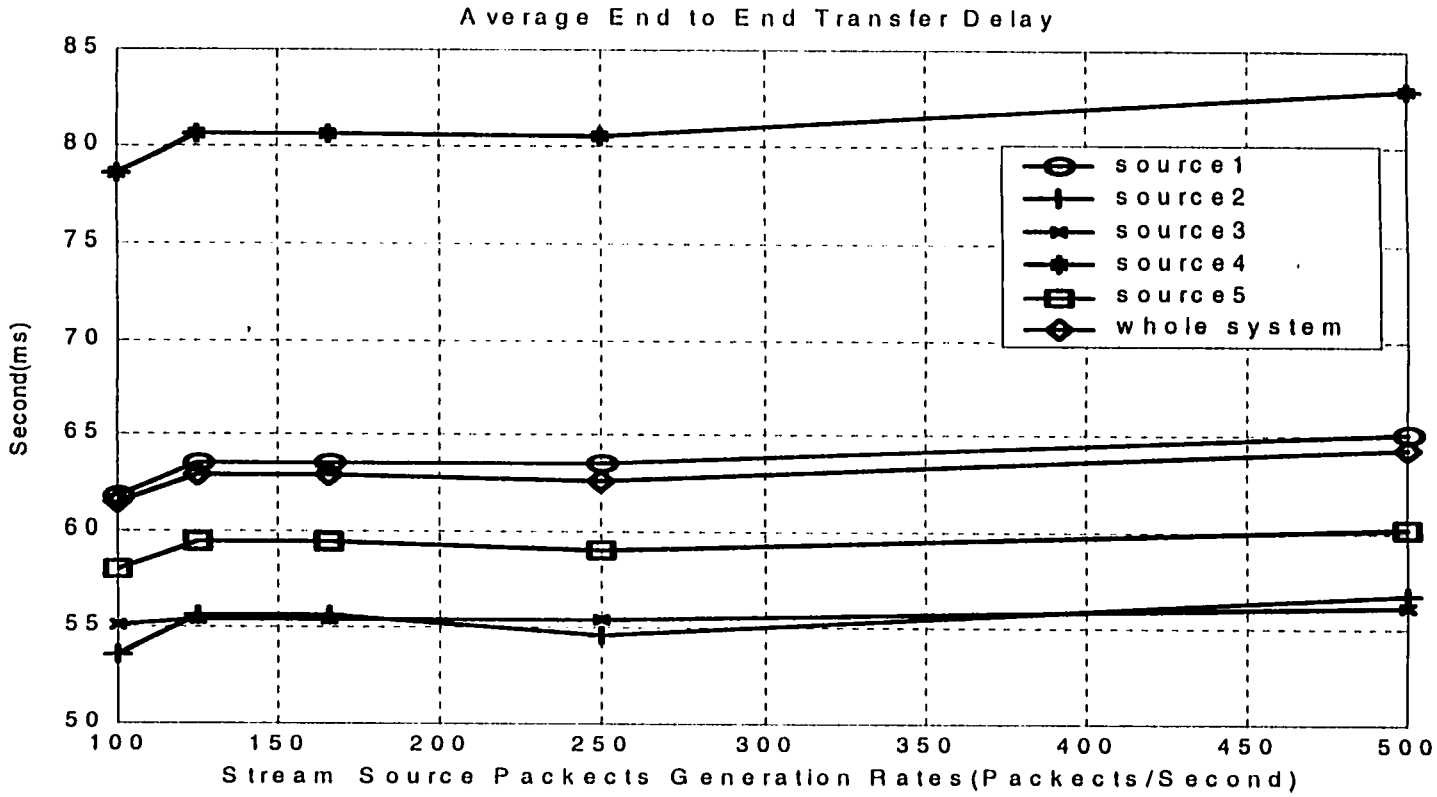


Figure 3-16: Average Total Transfer Delay vs. Packets Generation Rates
(ρ=1, Pc=0.99, Buffer Size =50, TTL=62, Simulation Time=1e6)

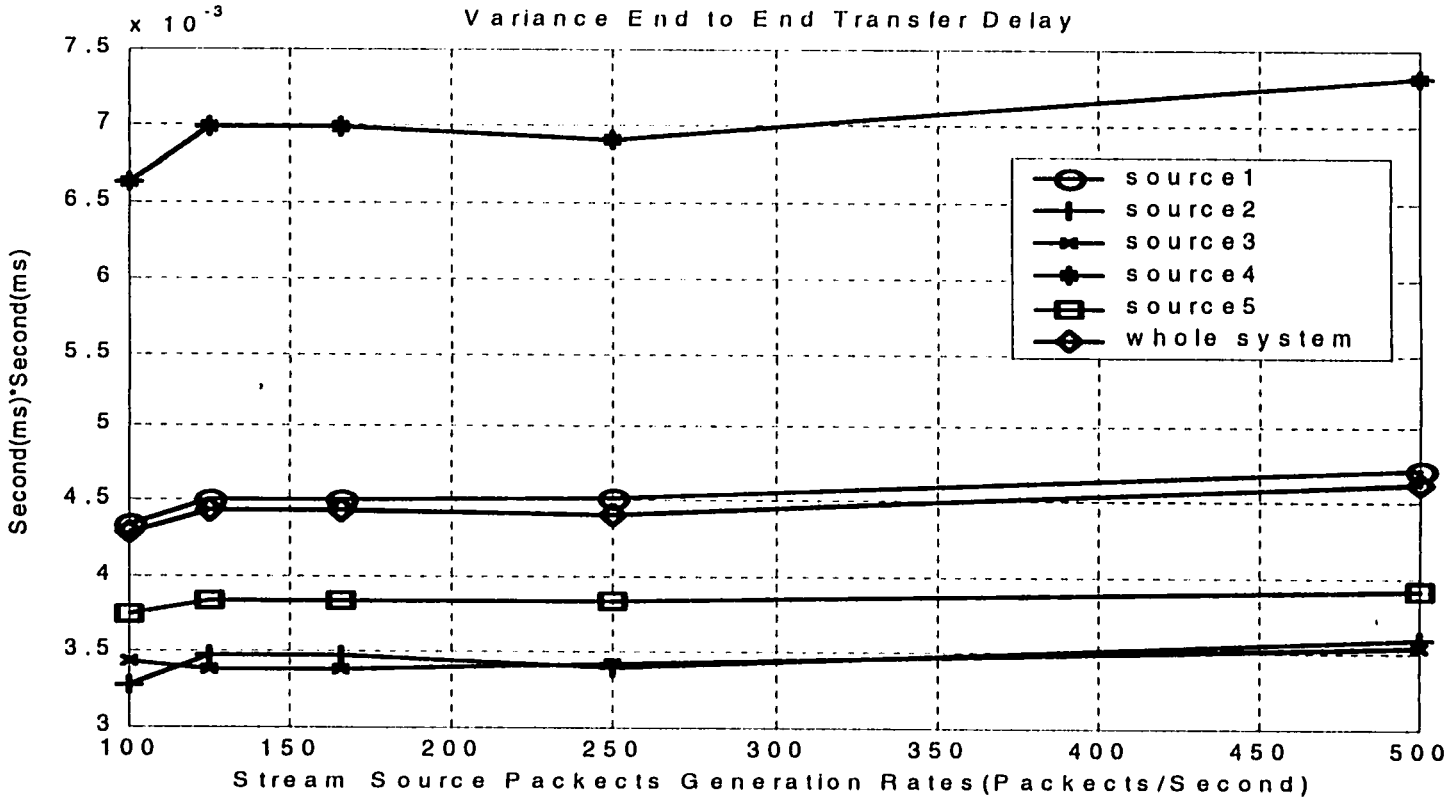


Figure 3-17: Variance Total Transfer Delay vs. Packets Generation Rates
(ρ=1, Pc=0.99, Buffer Size =50, TTL=62, Simulation Time=1e6)

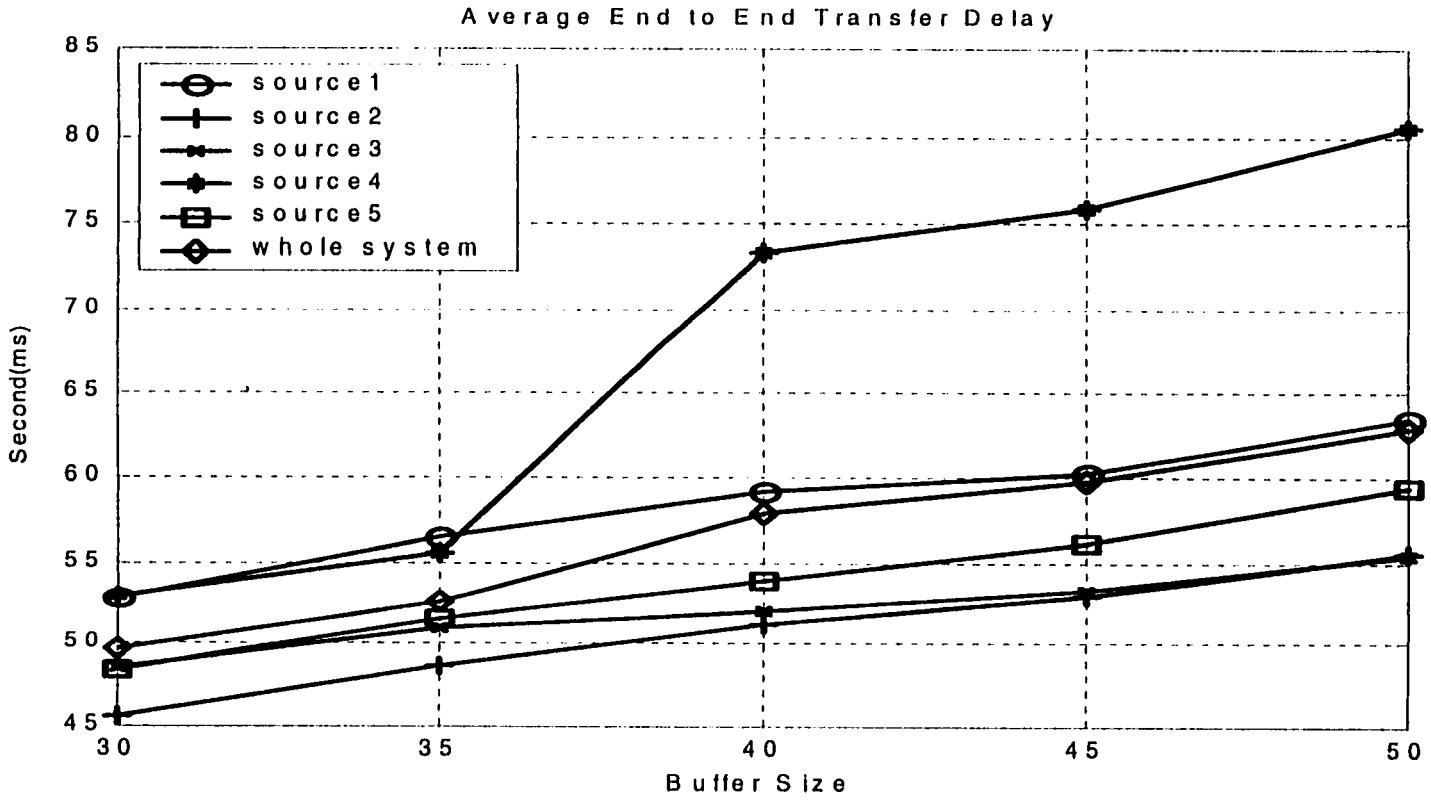


Figure 3-18:Average Total Transfer Delay vs. Buffer sizes
(ρ=1, Pc=0.99, Generation Rates=1000Packeets/Second, TTL=62, Simulation Time=1e6)

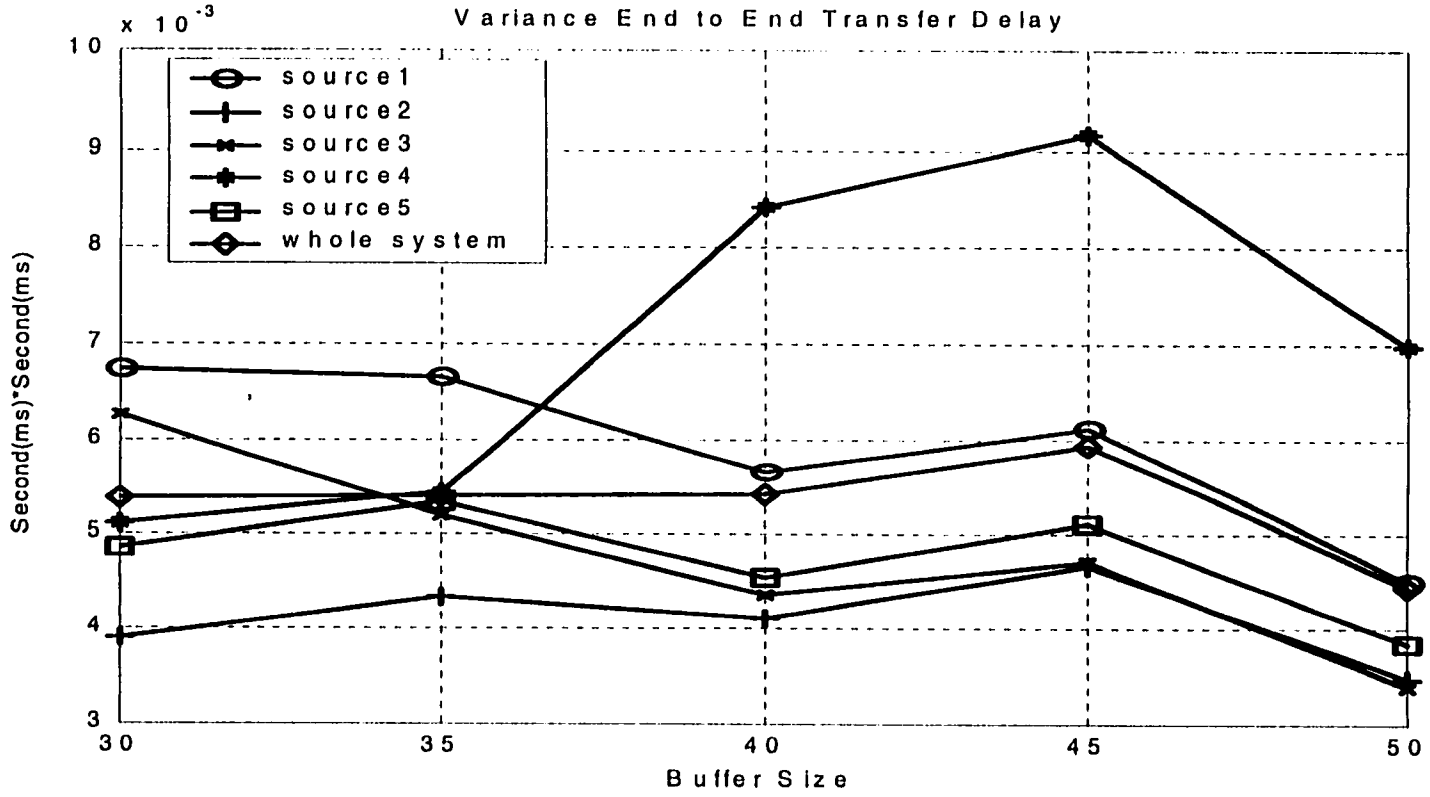


Figure 3-19:Variance Total Transfer Delay vs. Buffer Sizes
(ρ=1, Pc=0.99, Generation Rates=1000Packeets/Second, TTL=62, Simulation Time=1e6)

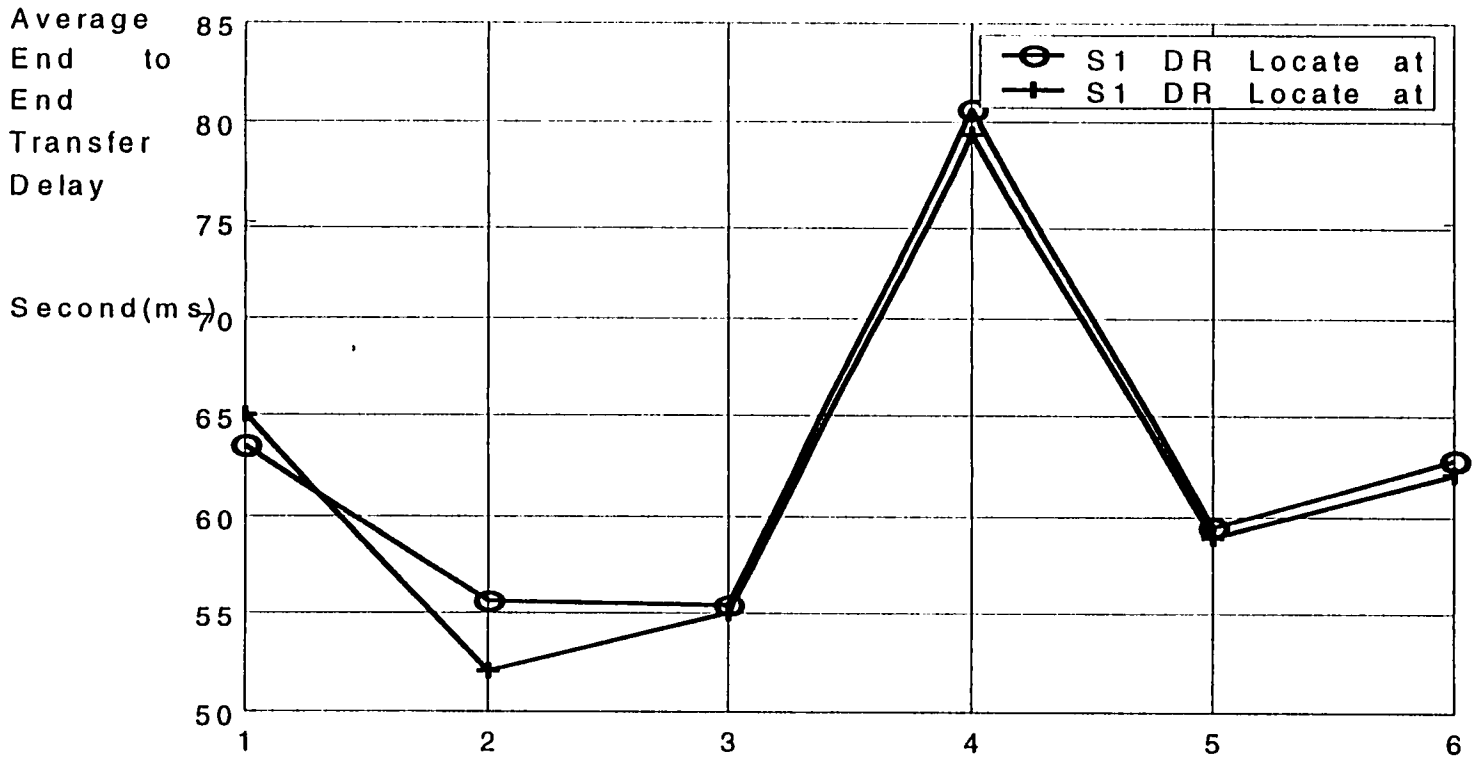


Figure 3-20: Average Total Transfer Delay vs. S1 DR location changing
(ρ=1 Pc=0.99, Generation Rates=1000Packeets/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

Source identity (1=S1 flow, 2=S2 flow, 3=S3 flow, 4=S4 flow, 5=S5 flow, 6=Average Results of all flows)

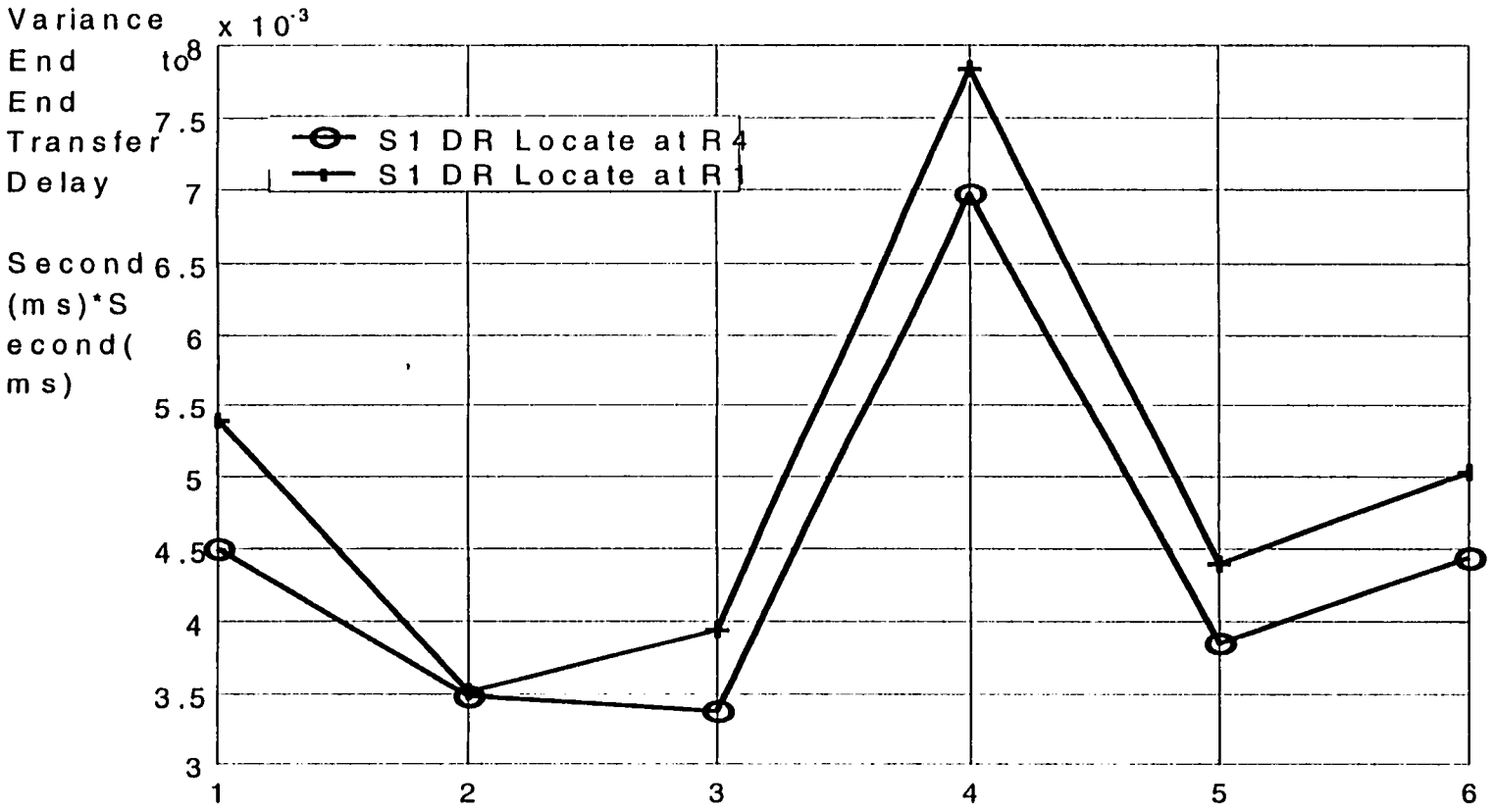


Figure 3-21: Variance Total Transfer Delay vs. S1 DR location changing
(ρ=1 Pc=0.99, Generation Rates=1000Packects/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

Source identity (1=S1 flow, 2=S2 flow, 3=S3 flow, 4=S4 flow, 5=S5 flow, 6=Average Results of all flows)

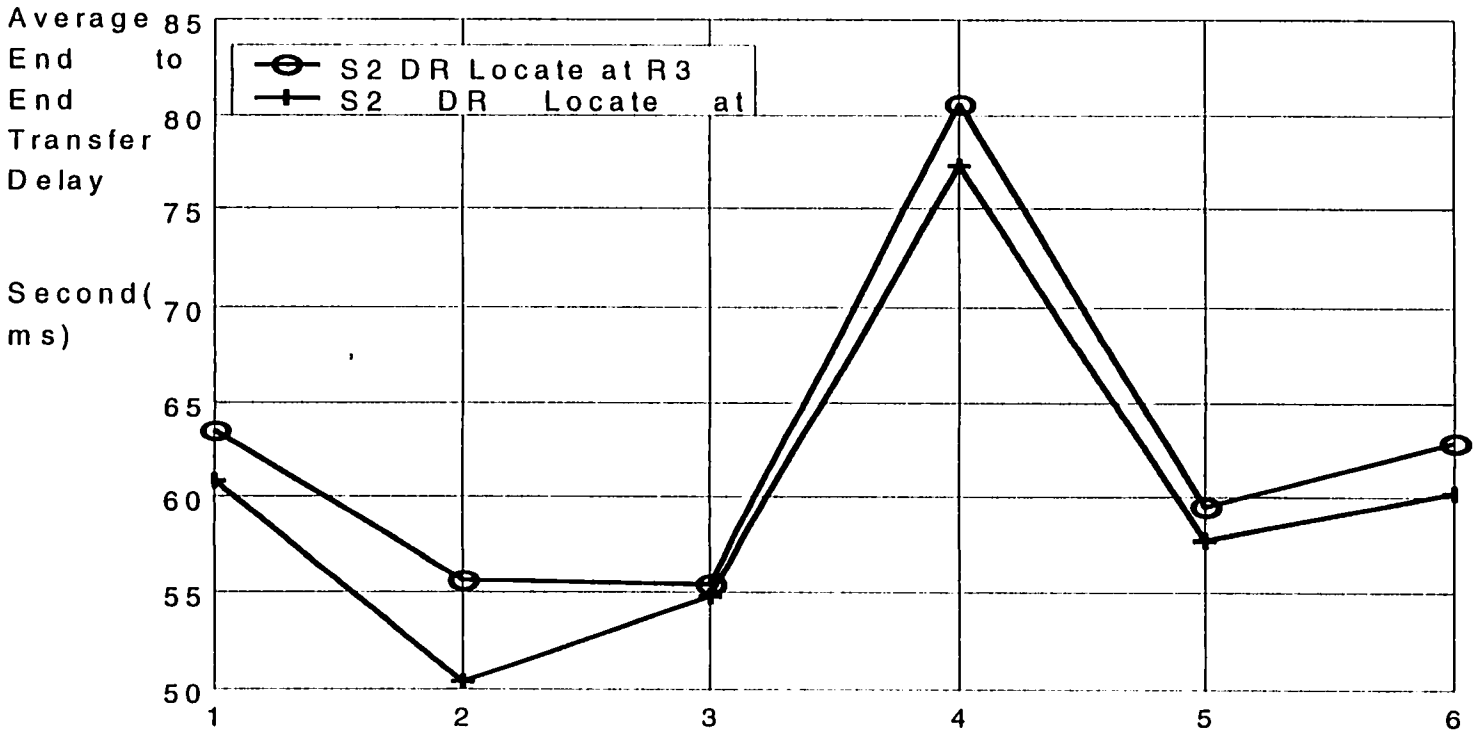


Figure 3-22: Average Total Transfer Delay vs. S2 DR location changing
(ρ=1 Pc=0.99, Generation Rates=1000Packects/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

Source identity (1=S1 flow, 2=S2 flow, 3=S3 flow, 4=S4 flow, 5=S5 flow, 6=Average Results of all flows)

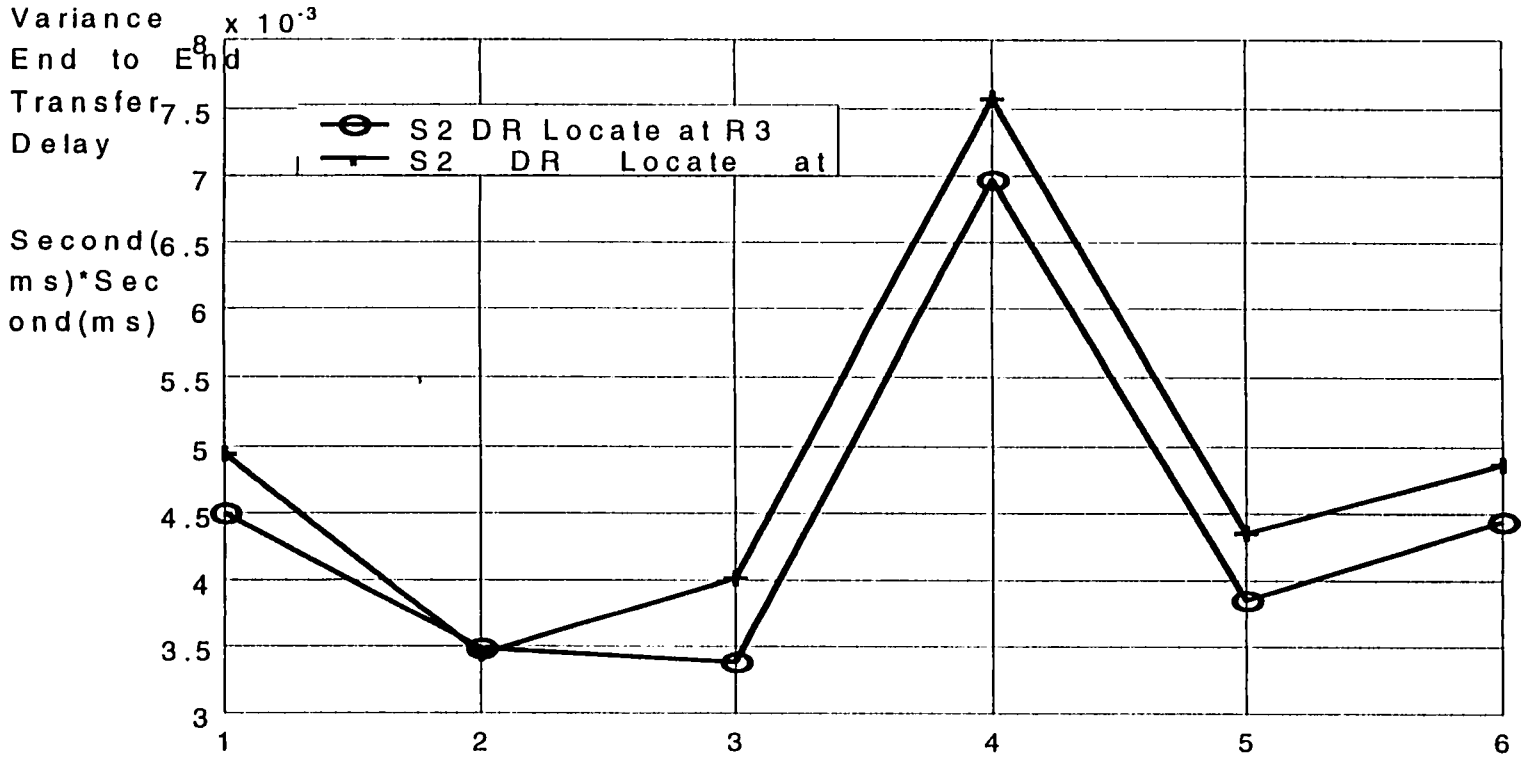


Figure 3-23: Variance Total Transfer Delay vs. S2 DR location changing
(ρ=1 Pc=0.99, Generation Rates=1000Packects/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

Source identity (1=S1 flow, 2=S2 flow, 3=S3 flow, 4=S4 flow, 5=S5 flow, 6=Average Results of all flows)

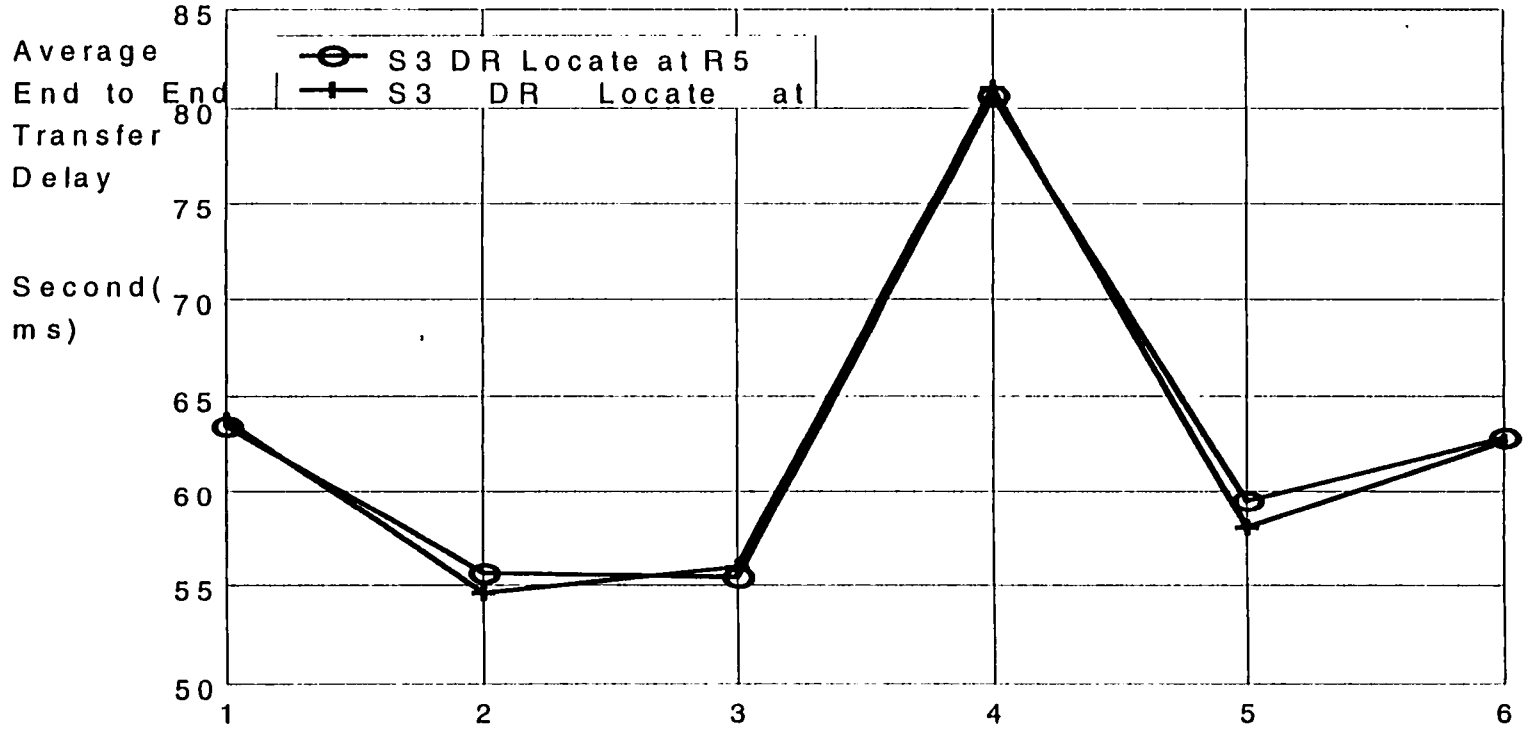


Figure 3-24: Average Total Transfer Delay vs. S3 DR location changing
(ρ=1 Pc=0.99, Generation Rates=1000Packects/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

Source identity (1=S1 flow, 2=S2 flow, 3=S3 flow, 4=S4 flow, 5=S5 flow, 6=Average Results of all flows)

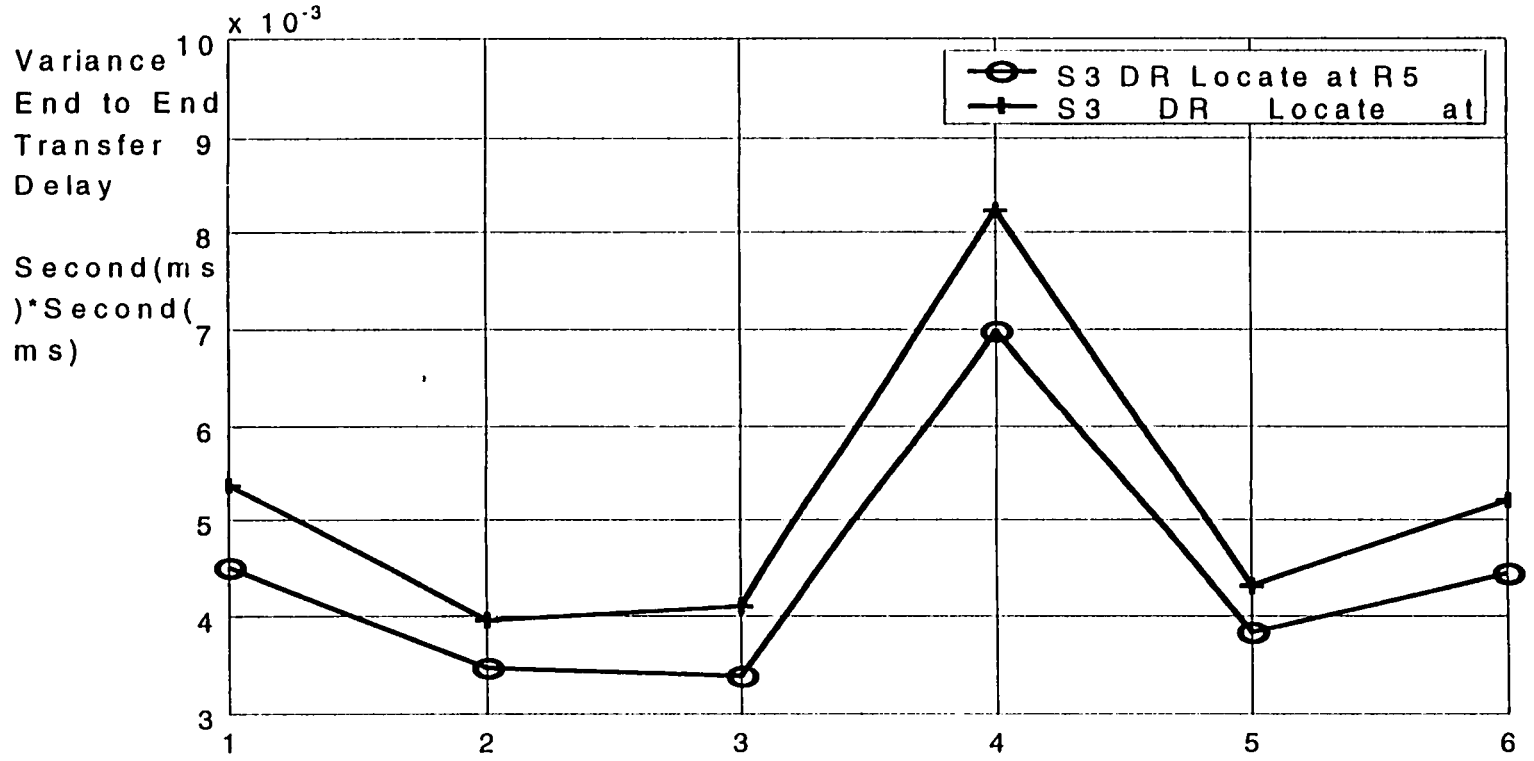


Figure 3-25: Variance Total Transfer Delay vs. S3 DR location changing
(ρ=1 Pc=0.99, Generation Rates=1000Packects/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

Source identity (1=S1 flow, 2=S2 flow, 3=S3 flow, 4=S4 flow, 5=S5 flow, 6=Average Results of all flows)

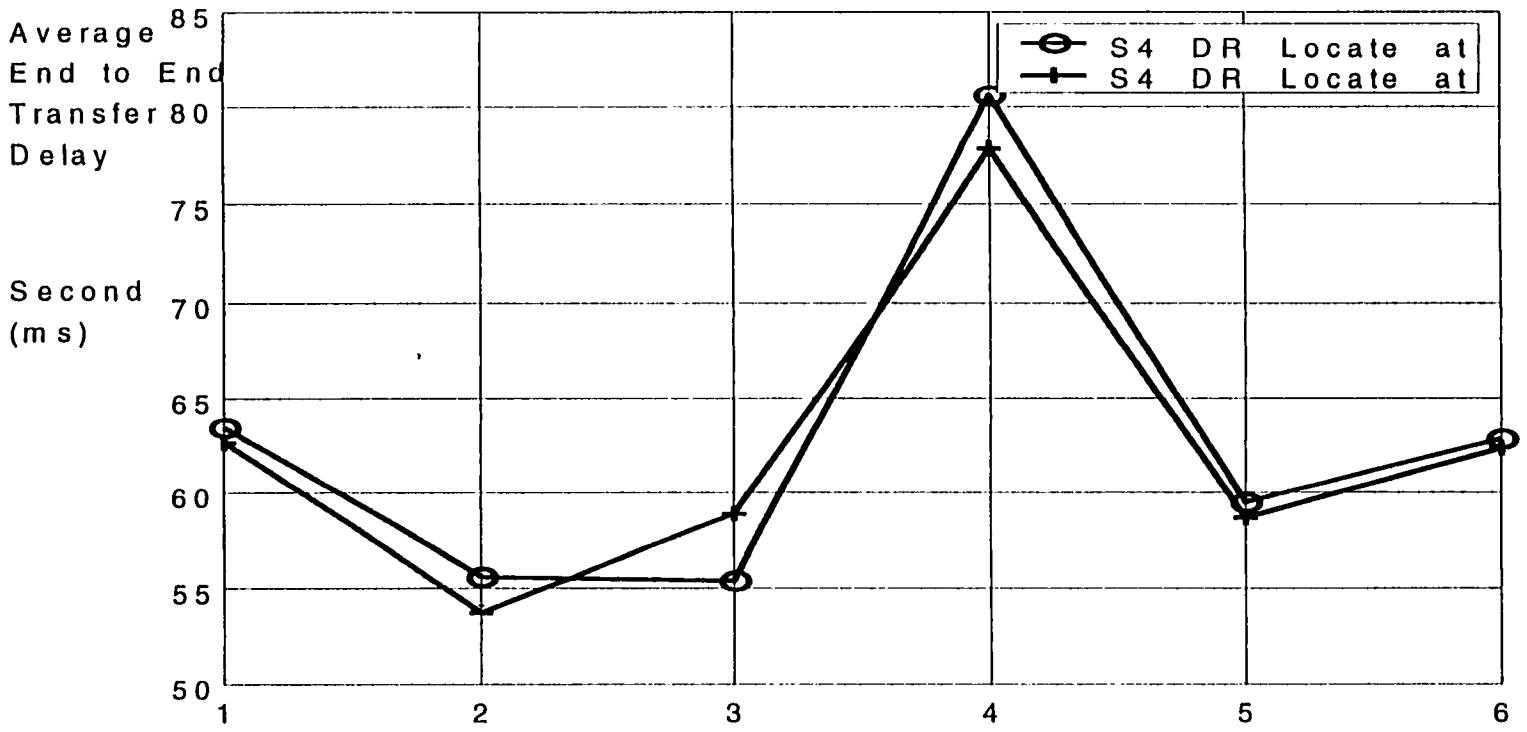


Figure 3-26: Average Total Transfer Delay vs. S4 DR location changing
(ρ=1 Pc=0.99, Generation Rates=1000Packects/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

Source identity (1=S1 flow, 2=S2 flow, 3=S3 flow, 4=S4 flow, 5=S5 flow, 6=Average Results of all flows)

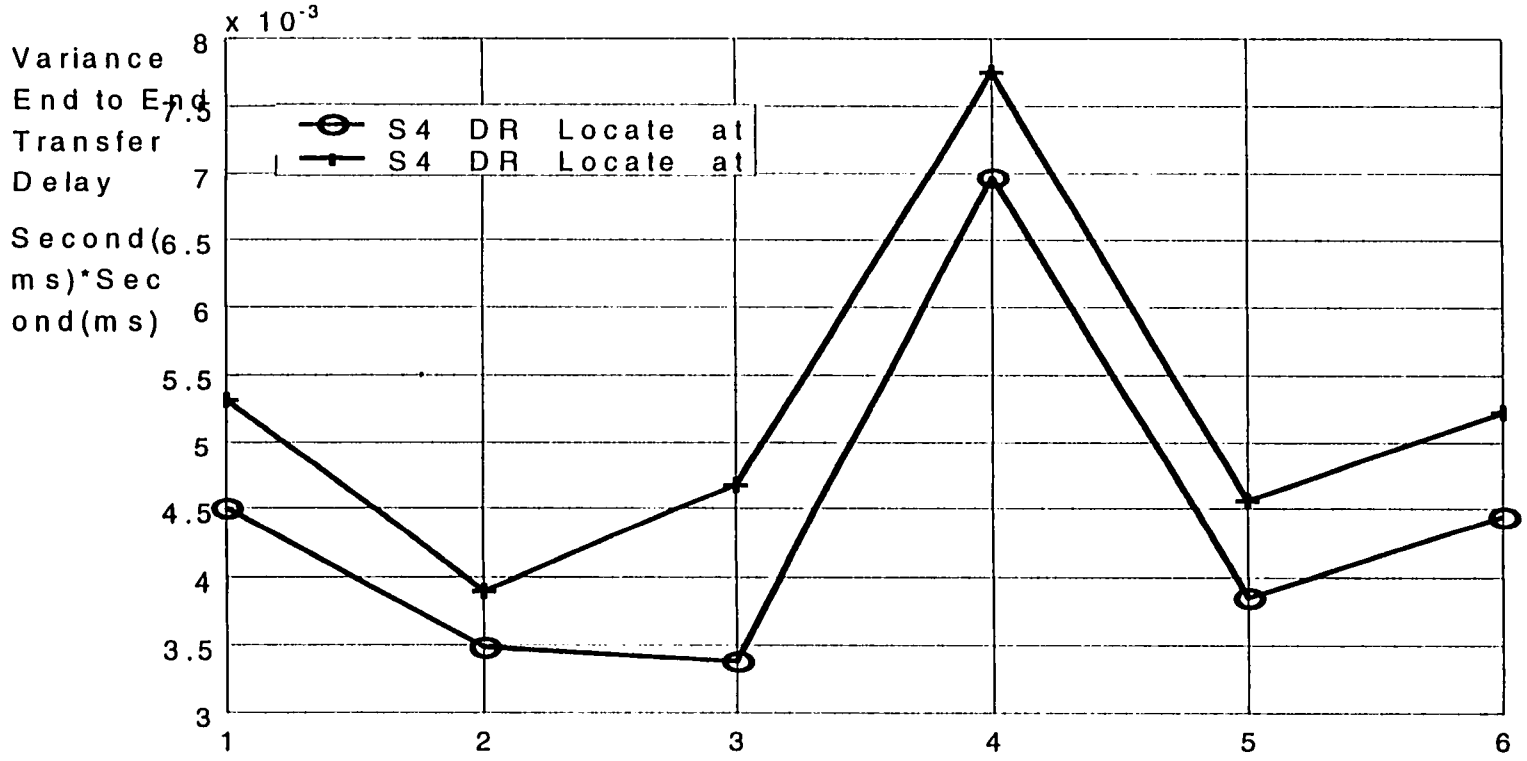


Figure 3-27: Variance Total Transfer Delay vs. S4 DR location changing
(ρ=1 Pc=0.99, Generation Rates=1000Packects/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

Source identity (1=S1 flow, 2=S2 flow, 3=S3 flow, 4=S4 flow, 5=S5 flow, 6=Average Results of all flows)

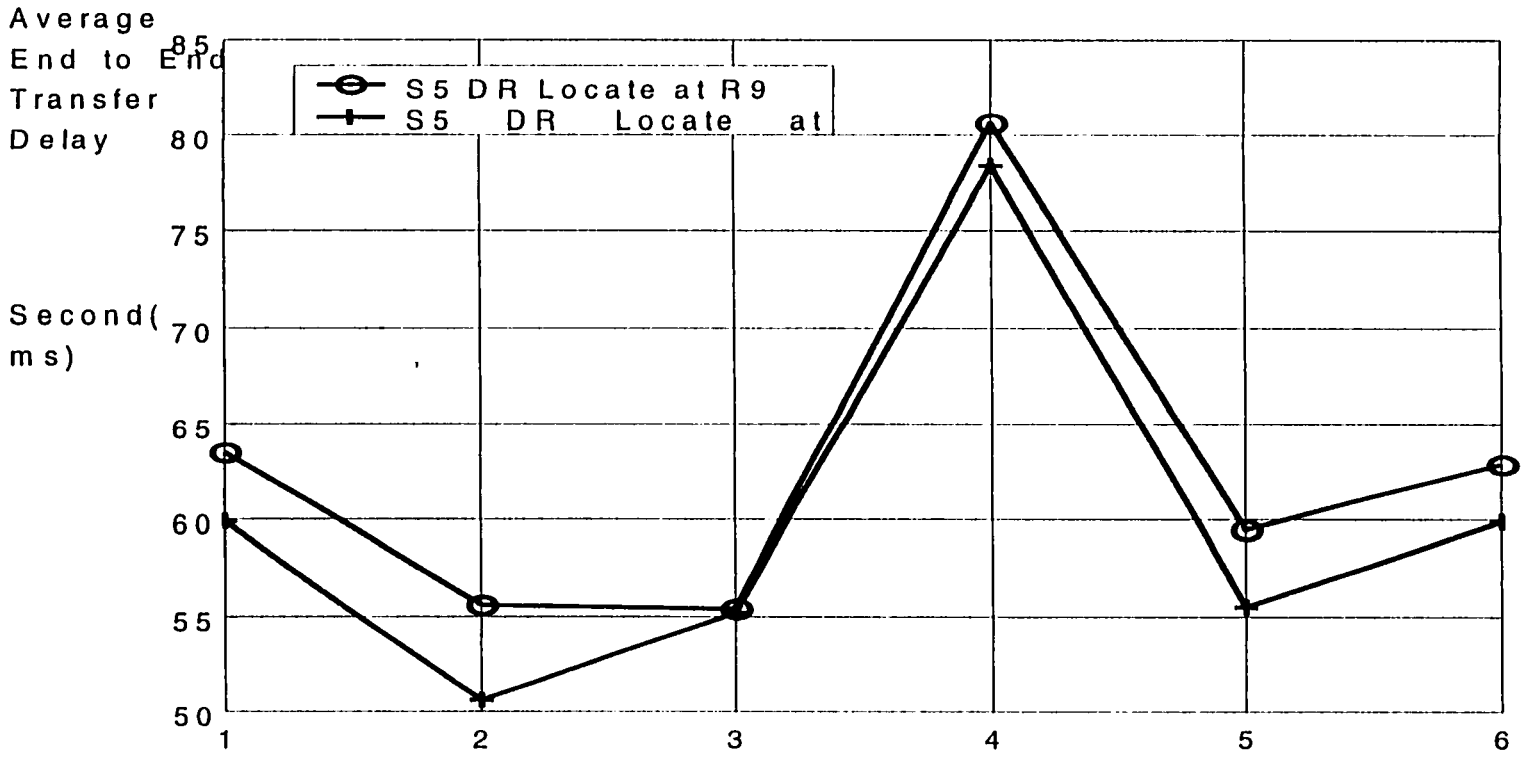


Figure 3-28: Average Total Transfer Delay vs. S5 DR location changing
(ρ=1 Pc=0.99, Generation Rates=1000Packects/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

Source identity (1=S1 flow, 2=S2 flow, 3=S3 flow, 4=S4 flow,5=S5 flow, 6=Average Results of all flows)

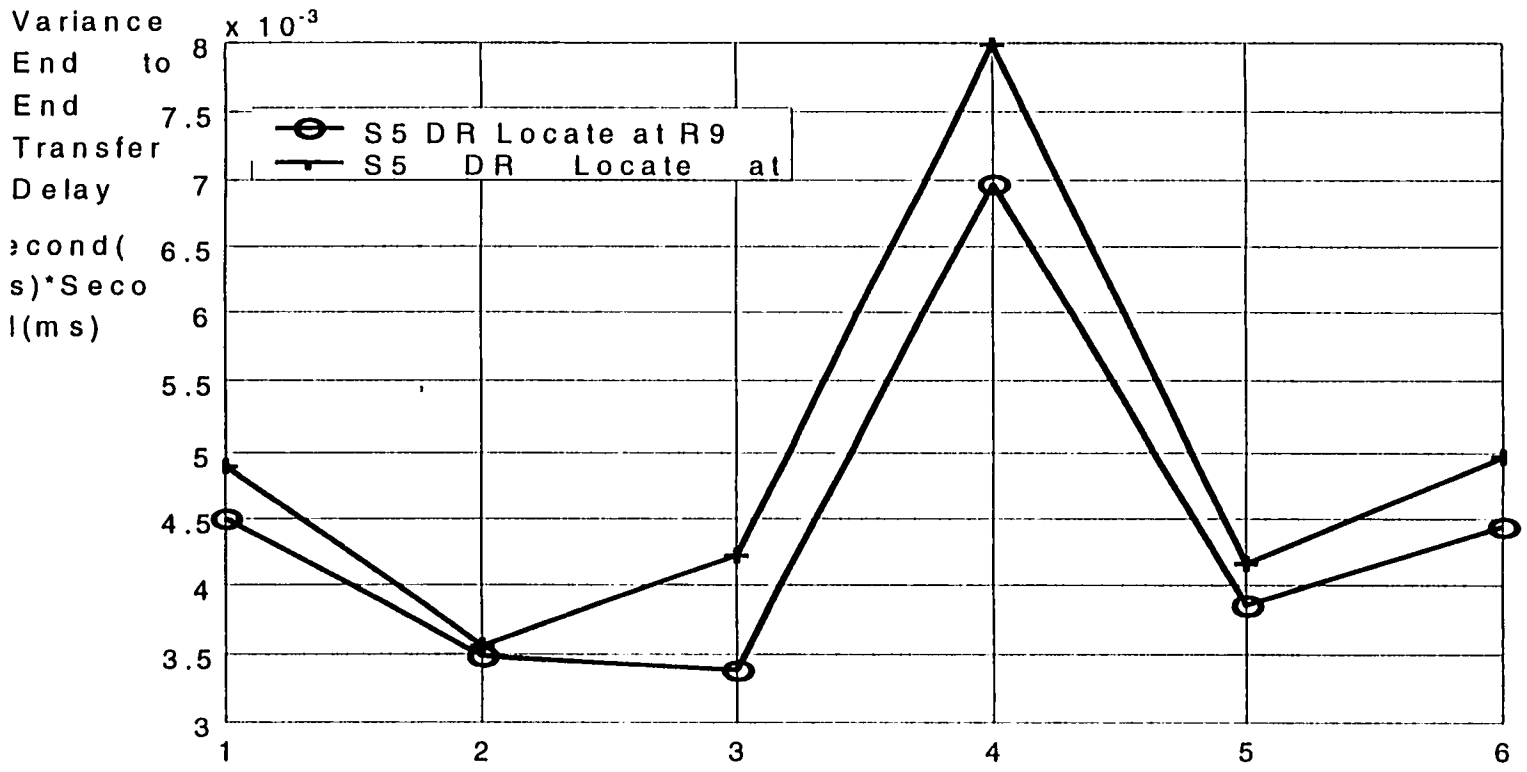


Figure 3-29: Variance Total Transfer Delay vs. S5 DR location changing
(ρ=1 Pc=0.99, Generation Rates=1000Packeets/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

Source identity (1=S1 flow, 2=S2 flow, 3=S3 flow, 4=S4 flow,5=S5 flow, 6=Average Results of all flows)

3.4.2 Average End-to-End Throughput (AT)

Average end-to-end throughput is a very important criterion; it is the ratio of the number of packets that are successfully transmitted in a very long interval to the maximum number of packets that could have been transmitted with continued transmission on the channel. As shown in Figure 3-30, we can see that the average throughput decreases by increasing the generation rates, because more traffic means less opportunity for packets to succeed.

However, the buffer size does not affect this criterion too much, since when enlarging the buffer size, the message has less of a chance to overflow□ but has more of a chance to suffer timeout. This is the trade-off between these two criteria. The result, however, is that too small or too large buffer sizes will decrease the average end-to-end throughput, as shown in Figure 3-31. In our simulation, buffer size equal to 40 is an optimal point.

If we only change the Domain Receiver for the source one flow, we can see the DR located at R1 yields less average end-to-end throughput than that located at R4 for the whole network, and for source flow one the performance is the same, as in figure 3-32.

If we only change the Domain Receiver for the source two flows, we can see the DR located at R2 or R7 yields less average end to end throughput than that located at R3 for the whole network, and for source flow two, there are large decreases in performance because more traffic in the network, as in figure 3-33.

If we only change the Domain Receiver for the source three flows, we can see the DR located at R8 or R9 yields less than the average end-to-end throughput than that

located at R5 for the whole network. For source flow three, there are large decreases in performance cause more traffic in the network, as in figure 3-34.

If we only change the Domain Receiver for the source four flows, we can see that the DR located at R8 yields less than the average end-to-end throughput located at R5 for the whole network. For source flow four, there are large decreased for performance that causes more traffic in the network, as in figure 3-35.

If we only change the Domain Receiver for the source five flow, we can see DR that the located at R6 or R5 yields less that the average end-to-end throughput located at R9 for the whole network, and for source flow five, there are large decrease in performance that cause more traffic in the network, as in figure 3-36.

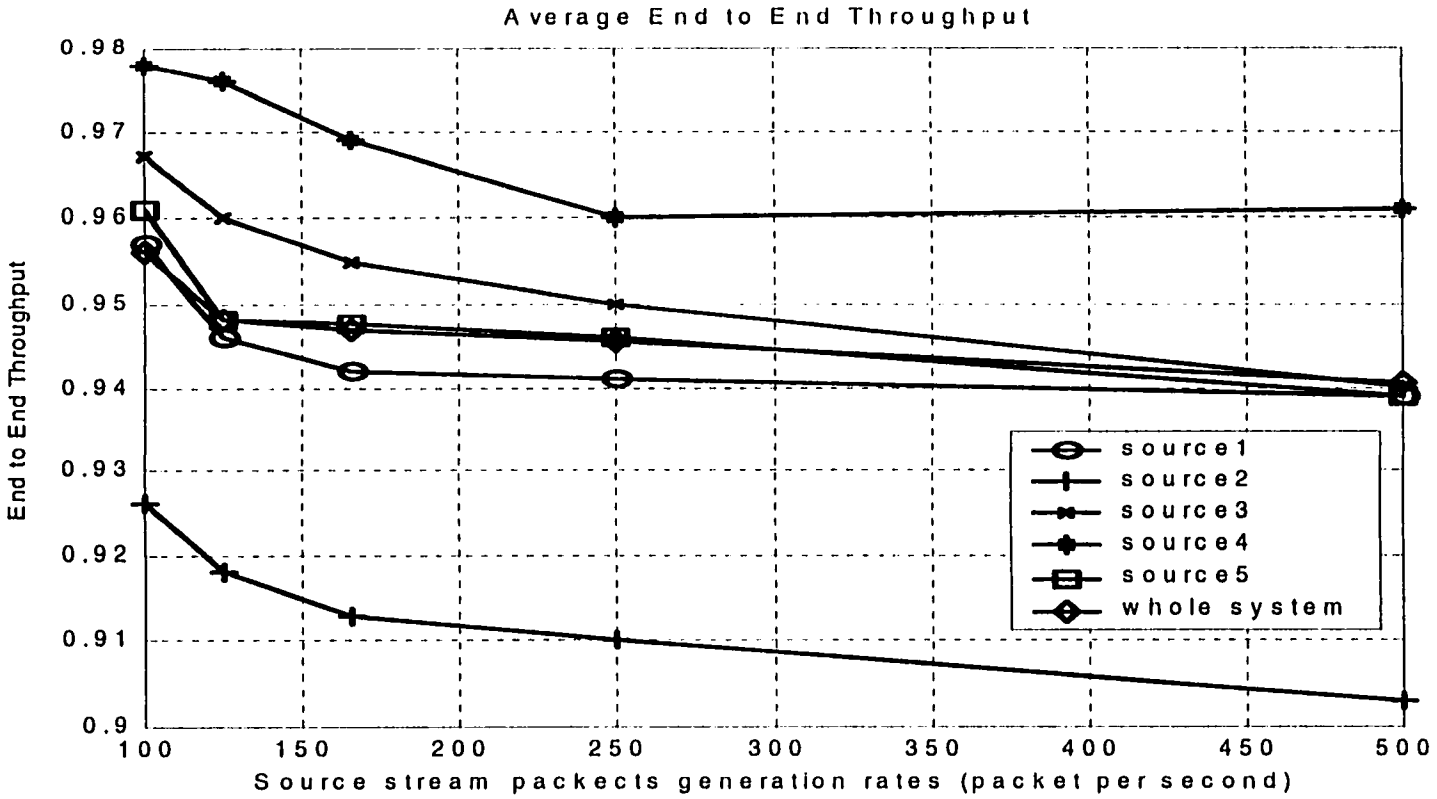


Figure 3-30: Average End to End Throughput vs. Packets Generation Rates
(ρ=1, Pc=0.99, Buffer Size =50, TTL=62, Simulation Time=1e6)

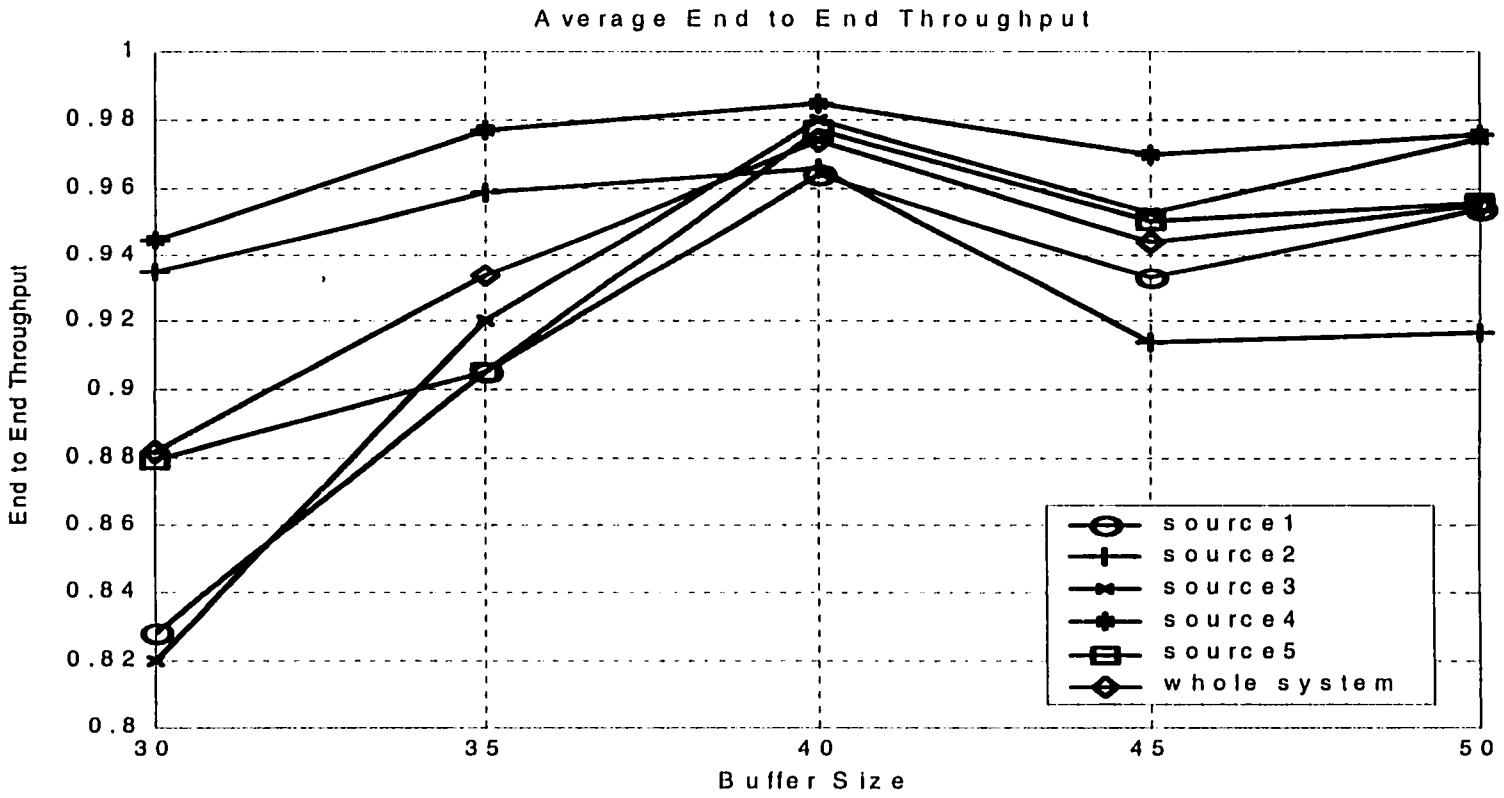


Figure 3-31: Average End to End Throughput vs. Buffer Size
(ρ=1, Pc=0.99, Generation Rates=1000Packeets/Second, TTL=62, Simulation Time=1e6)

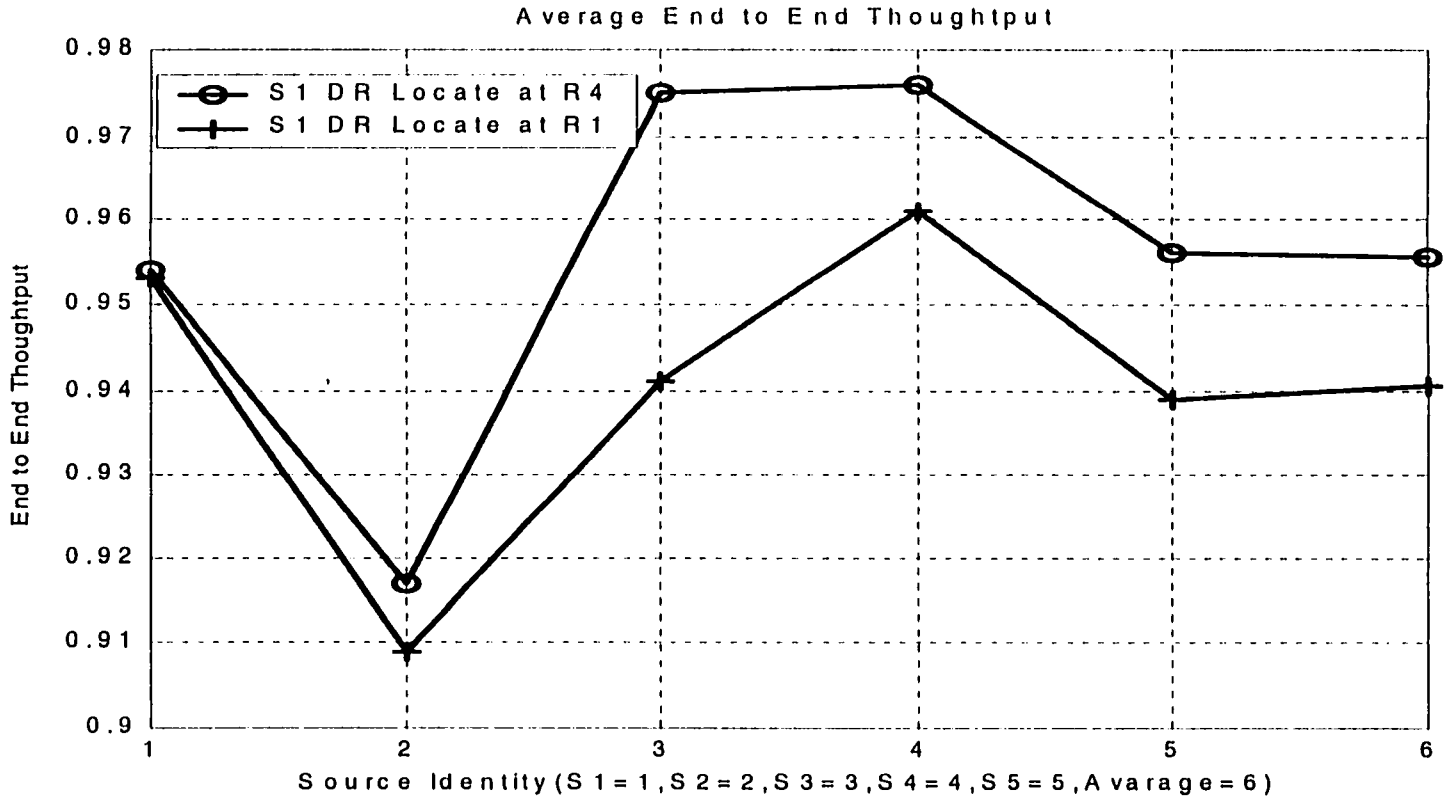


Figure 3-32: Average End to End Throughput vs. S1 DR location changing
(ρ=1 Pc=0.99, Generation Rates=1000Packects/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

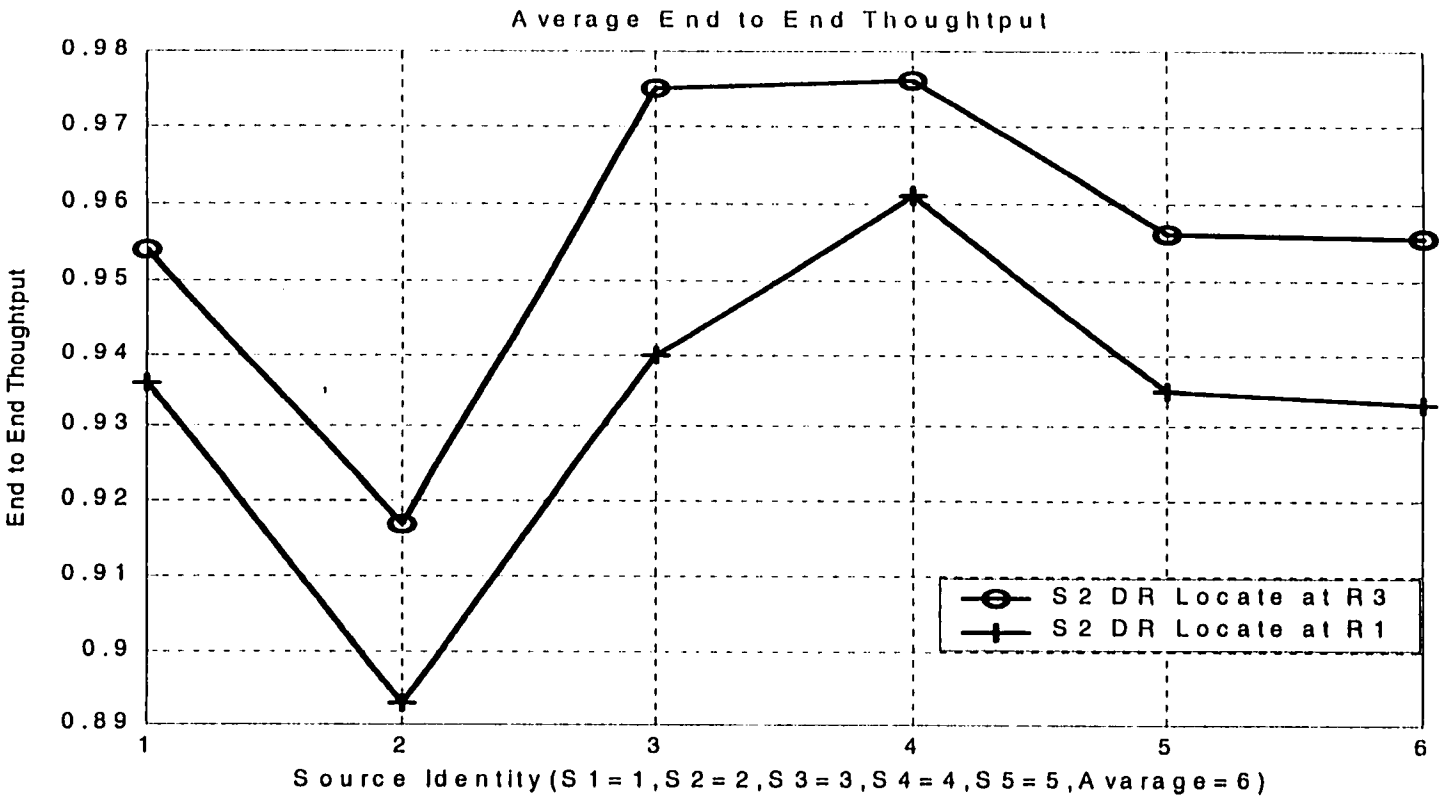


Figure 3-33: Average End to End Throughput vs. S2 DR location changing
(ρ=1 Pc=0.99, Generation Rates=1000Packects/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

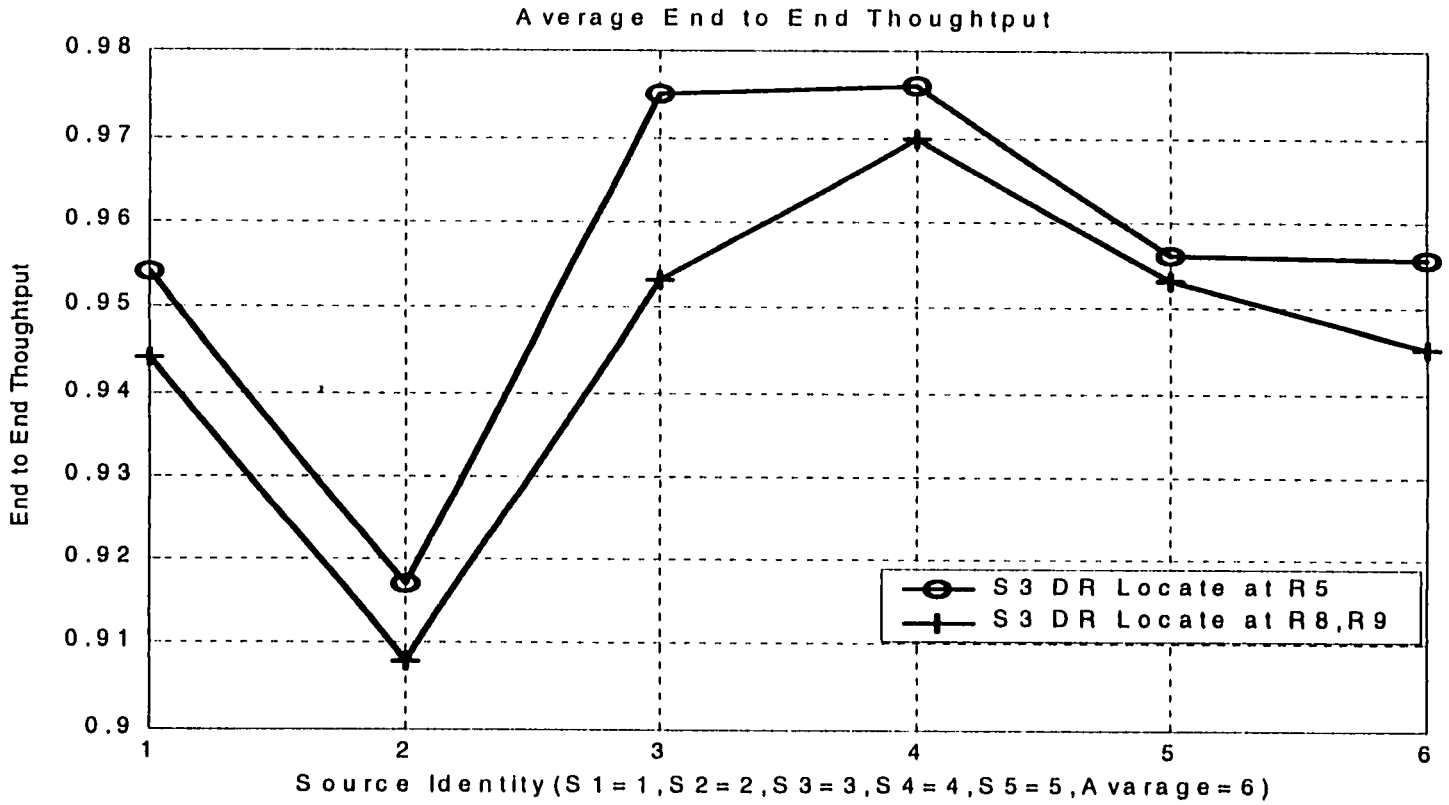


Figure 3-34: Average End to End Throughput vs. S3 DR location changing
(ρ=1 Pc=0.99, Generation Rates=1000Packects/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

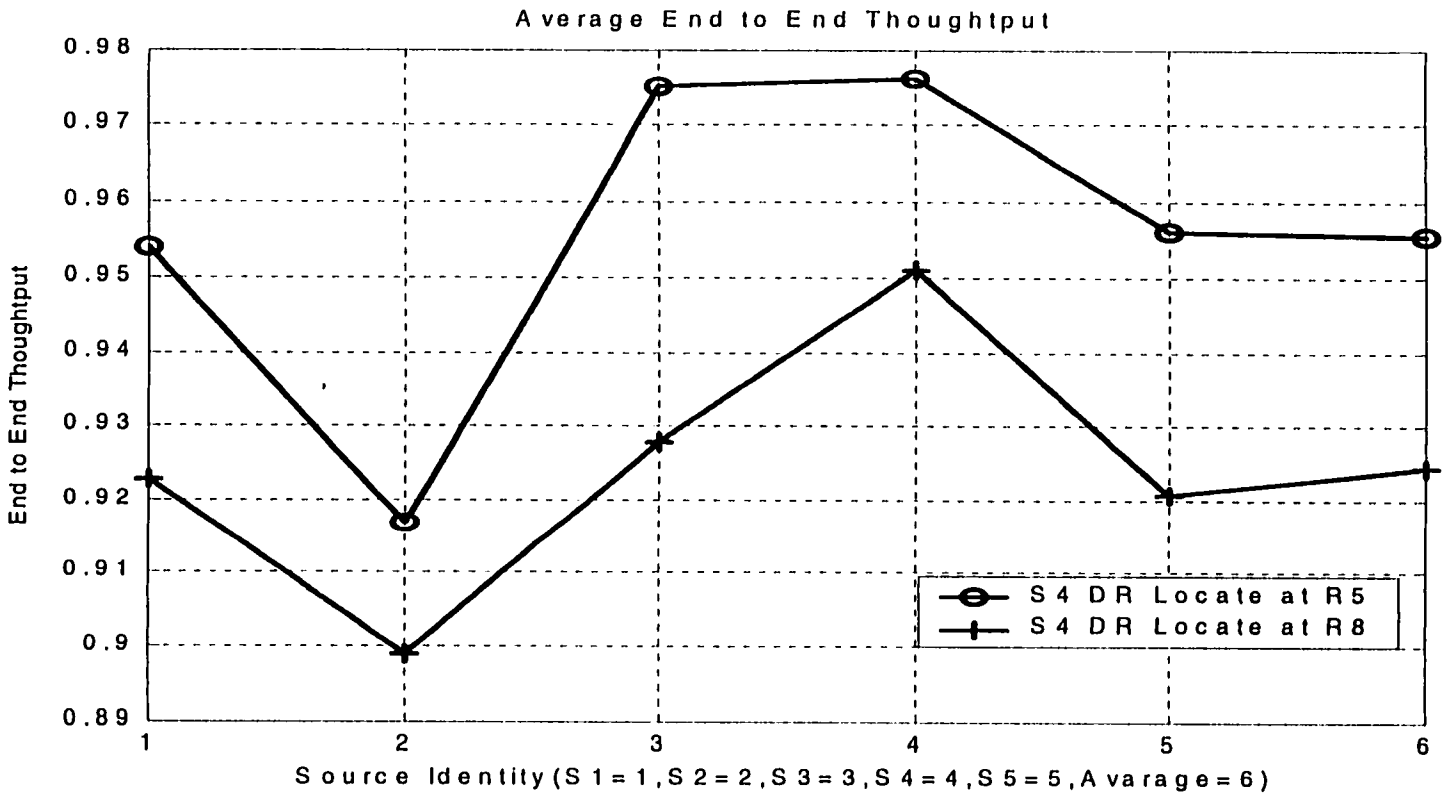


Figure 3-35: Average End to End Throughput vs. S4 DR location changing
(ρ=1 Pc=0.99, Generation Rates=1000Packects/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

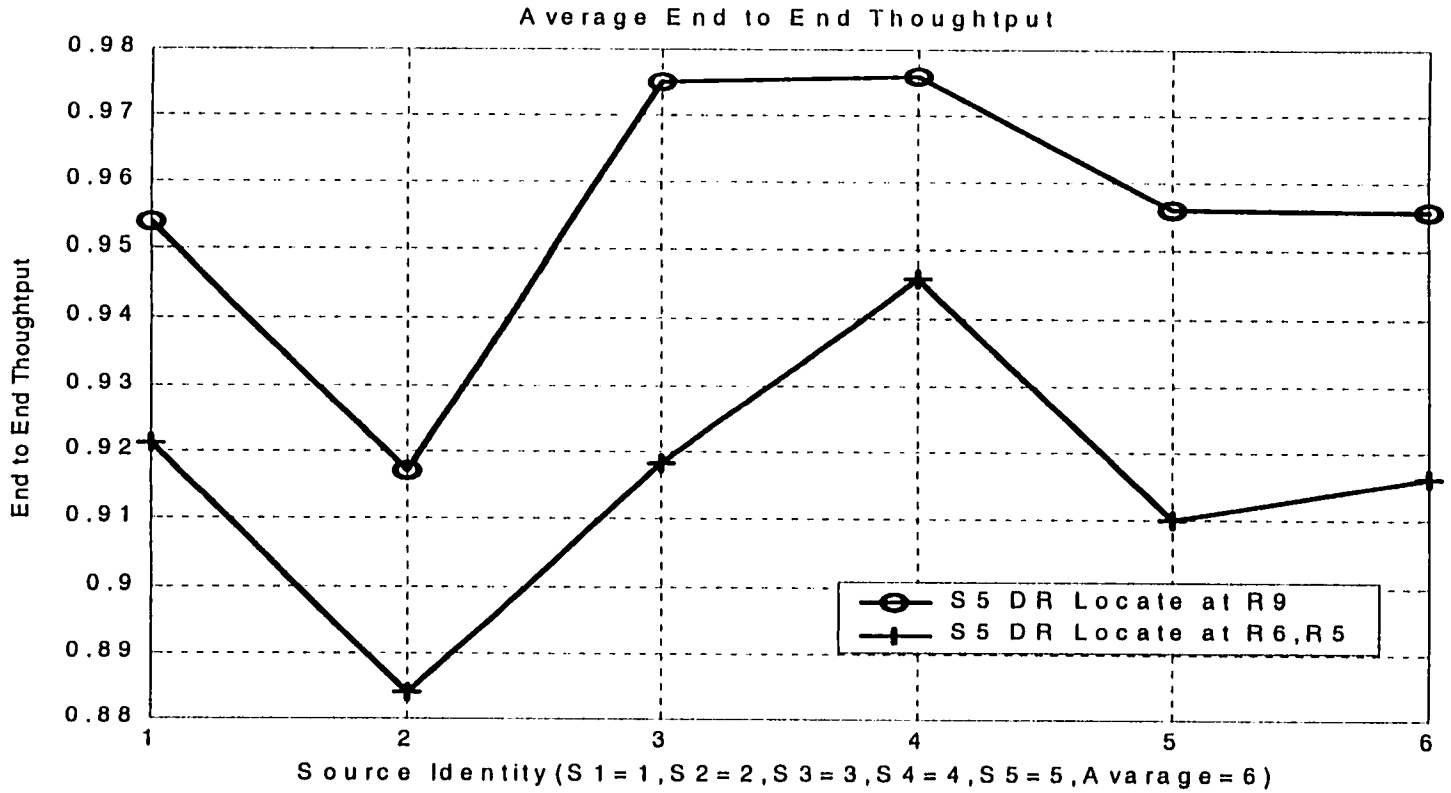


Figure 3-36: Average End to End Throughput vs. S5 DR location changing
(ρ=1 Pc=0.99, Generation Rates=1000Packects/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

3.4.3 Average Buffer Overflow (ABO)

This performance criterion is directly related to the average number of messages in a node buffer. It is seen that when the buffer size is small, there is more chance for the buffer to overflow, when the buffer size is gradually increased, less messages will encounter overflow, but the variance doesn't change too much.

As show in Figure 3-39, we fix the buffer size and the successful receiving probability (Pc) and the DR position and vary the source stream packets generation rates; after doing this; we can see the change of the average buffer overflow. Simulations are run by varying the generation rates from 100, 125, 166, 250, to 500 packets per second and fixing the buffer size at 50 packets, Timeout =62, Pc=0.99 and maximum number of iteration =1e6. It is seen that when the generation rates are low, the average buffer overflow is low, when the input traffic rise, the average buffer overflow also increases and the variance doesn't change much, as in figure 3-40.

Similarly, keeping the generation rates at 1000 packets per second, we run the simulation as buffer size varies. I In case of buffer size become too small, the packets have an overflow for limited buffer space, but as buffer size becomes too large, the packets will spend a long time queuing, in this case, some packets have more chance for time out, and are dropped from buffer. So a buffer size equal to 40 is an optimal point in our simulation. We run the simulation as buffer sizes vary from 30, 35, 40, 45, 50, and fix other parameters. It is seen that when the buffer size is small, the average buffer overflow is large, and when the buffer sizes increase, the average buffer overflow decreases. And the variance decreases as buffer sizes decrease, as in figure 3-37, 38.

Next, we change the Domain Receiver position one by one.

If we only change the Domain Receiver for source one, we can see that the DR located at R1 yields less than the average transfer delay that located at R4 for whole network. The variance DR located at R4 is better than R1, and for source flow one, at R1, there are large decreases in performance because it close to destinations, as in figure 3-41,42.

If we only change the Domain Receiver for source two, we can see that the DR located at R2, R7 yields less than the average buffer overflow that located at R3 for the whole network, and for source flow two, there are large improvements in performance because it is close to destinations. The variance doesn't change much, as in figure 3-43, 44.

If we only change the Domain Receiver for source three, we can see that the DR located at R8, R9 yields less than the average buffer overflow located at R5 for the whole network, and for source flow three, there are large improvements in performance because it is close to destinations. The variance doesn't change much, as in figure 3-45, 46.

If we only change the Domain Receiver for source four, we can see that the DR located at R8 yields less than the average buffer overflow located at R5 for the whole network, and for source flow four, there are large improvements in the performance because it is close to destinations. The variance doesn't change much, as in figure 3-47, 48.

If we only change the Domain Receiver for source five, we can see that the DR locate at R6, R5 yields less than the average buffer overflow located at R9 for the

whole network, and for source flow five, there are large improvement in the performance because it is close to destinations. The variance doesn't change much, as in figure 3-49, 50.

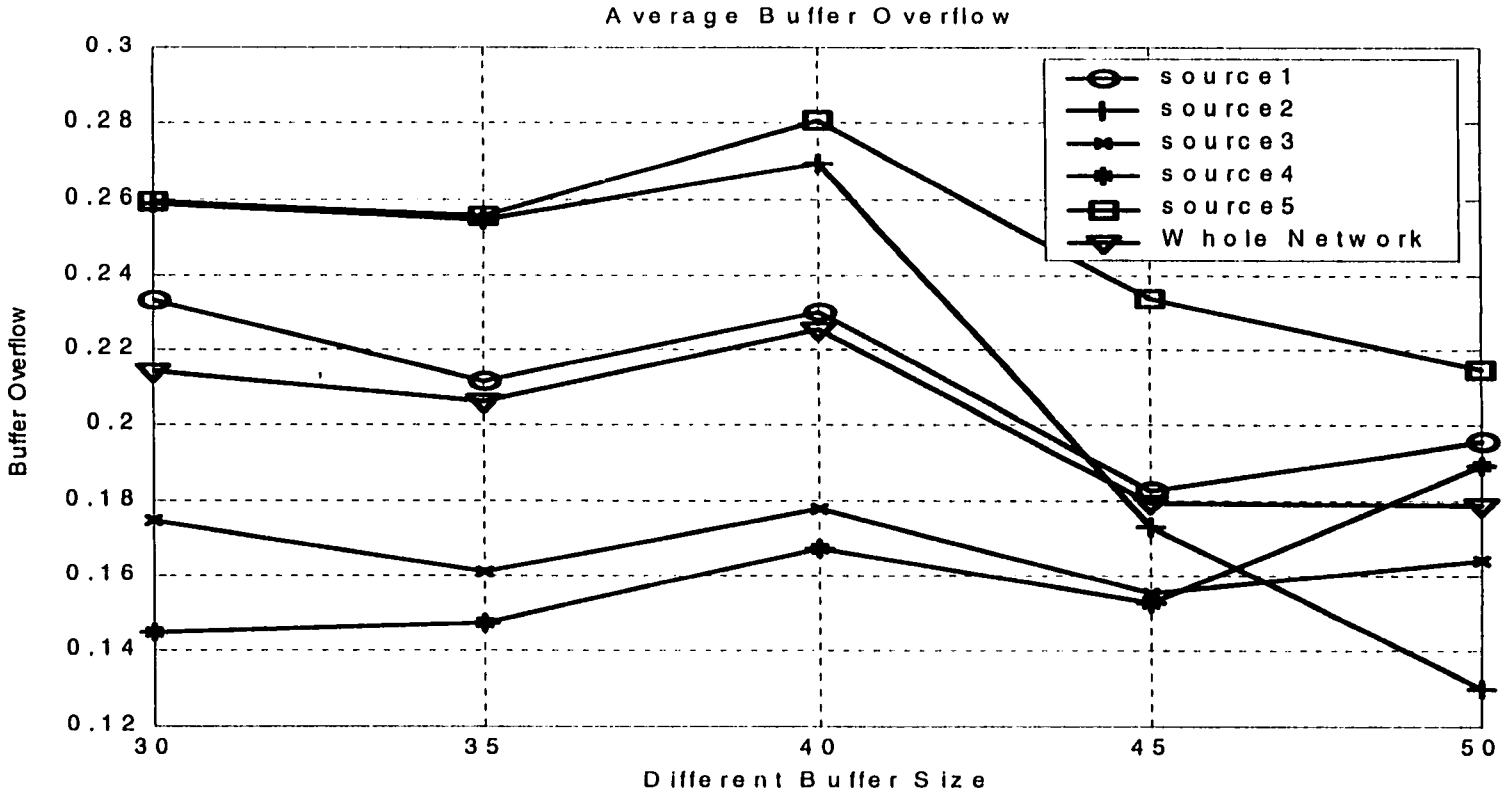


Figure 3-37: Average Buffer Overflow vs. Buffer Size
(ρ=1Pc=0.99, Generation Rates=1000Packects/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

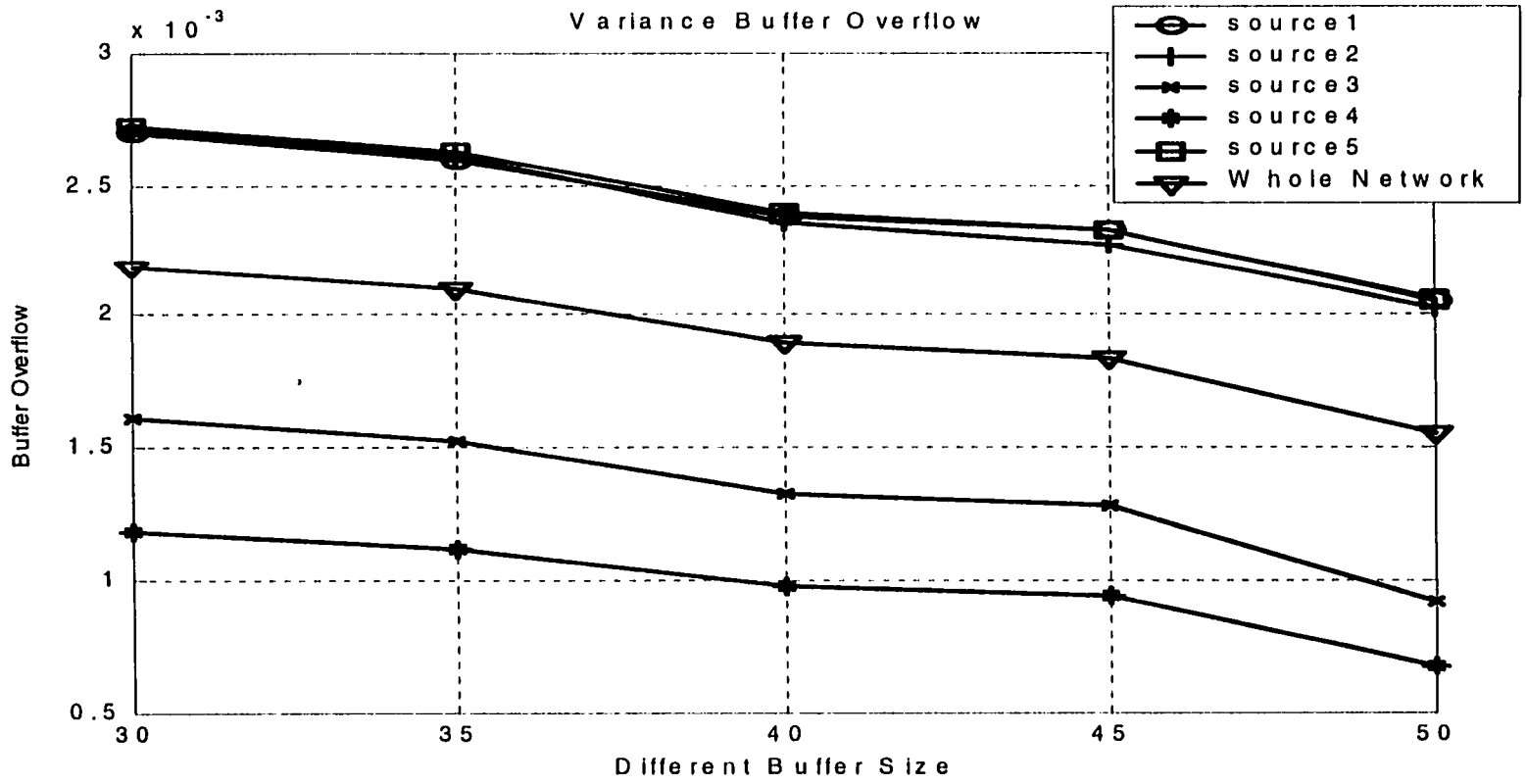


Figure 3-38: Variance overflow vs. Buffer Size
(ρ=1 Pc=0.99, Generation Rates=1000Packeets/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

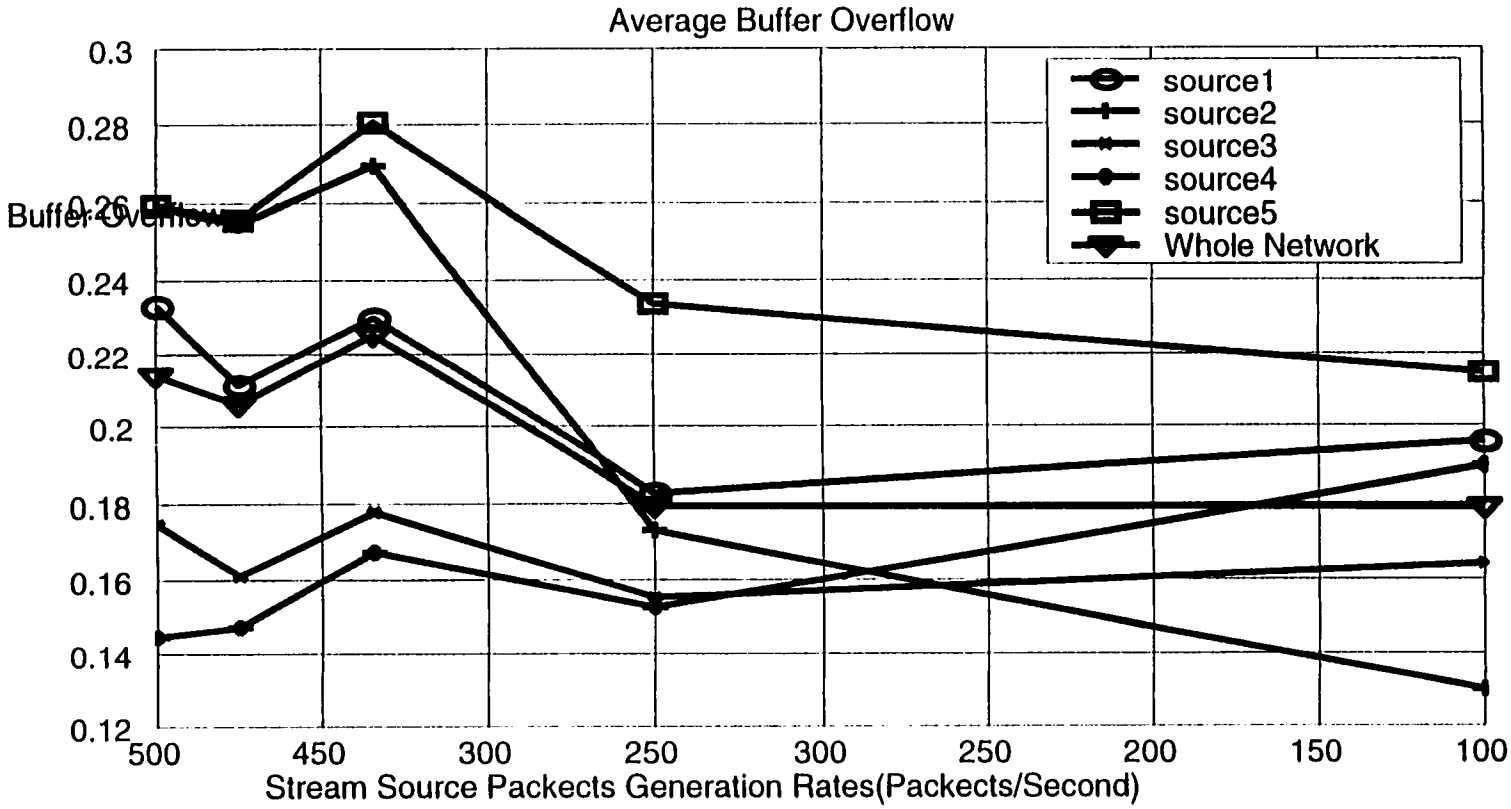


Figure 3-39: Average Buffer Overflow vs. Packets Generation Rates
(ρ=1, Pc=0.99, Generation Rates=1000Packects/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

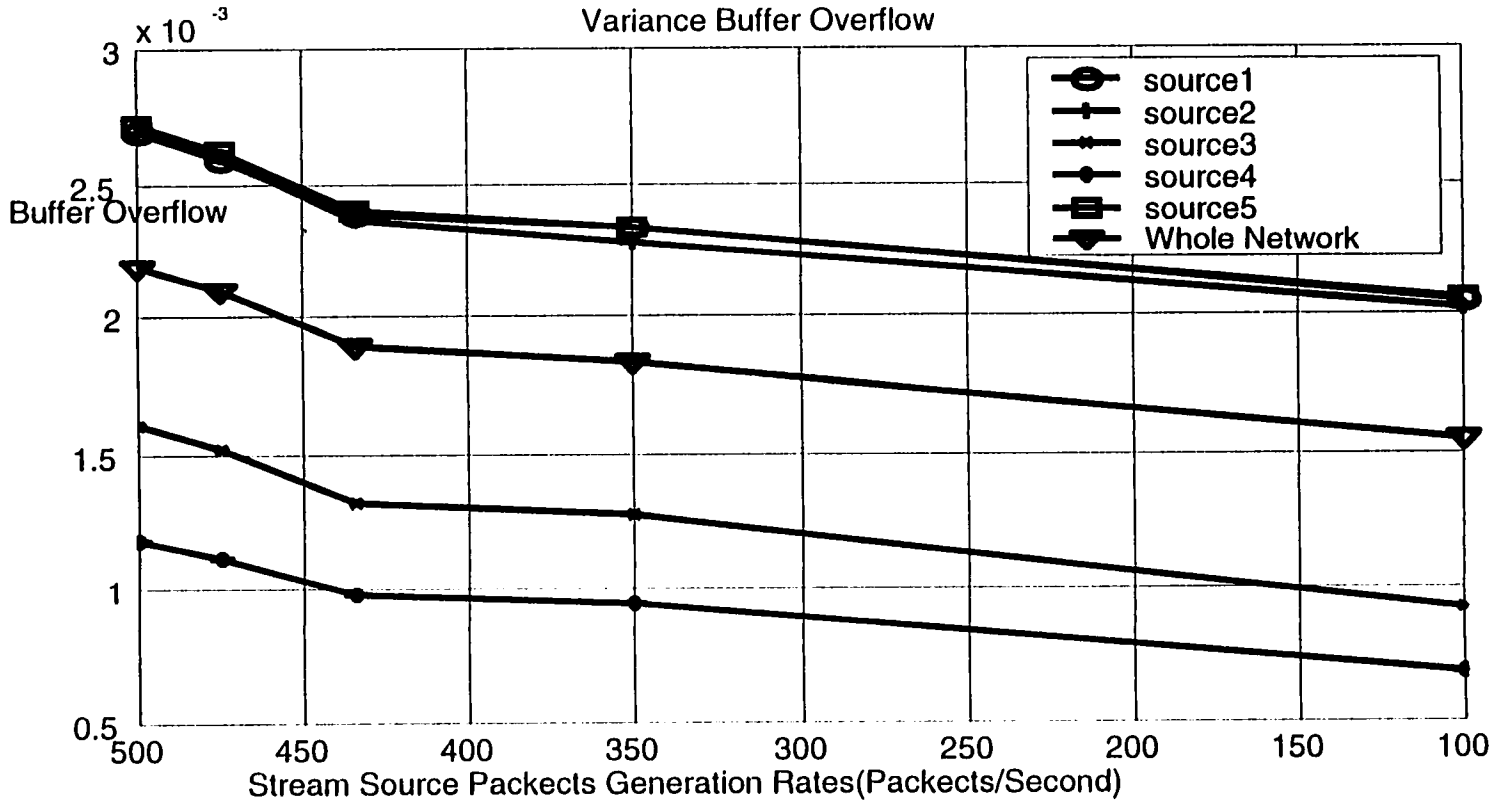


Figure 3-40: Variance Overflow vs. Packets Generation Rates
(ρ=1, Pc=0.99, Generation Rates=1000Packects/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

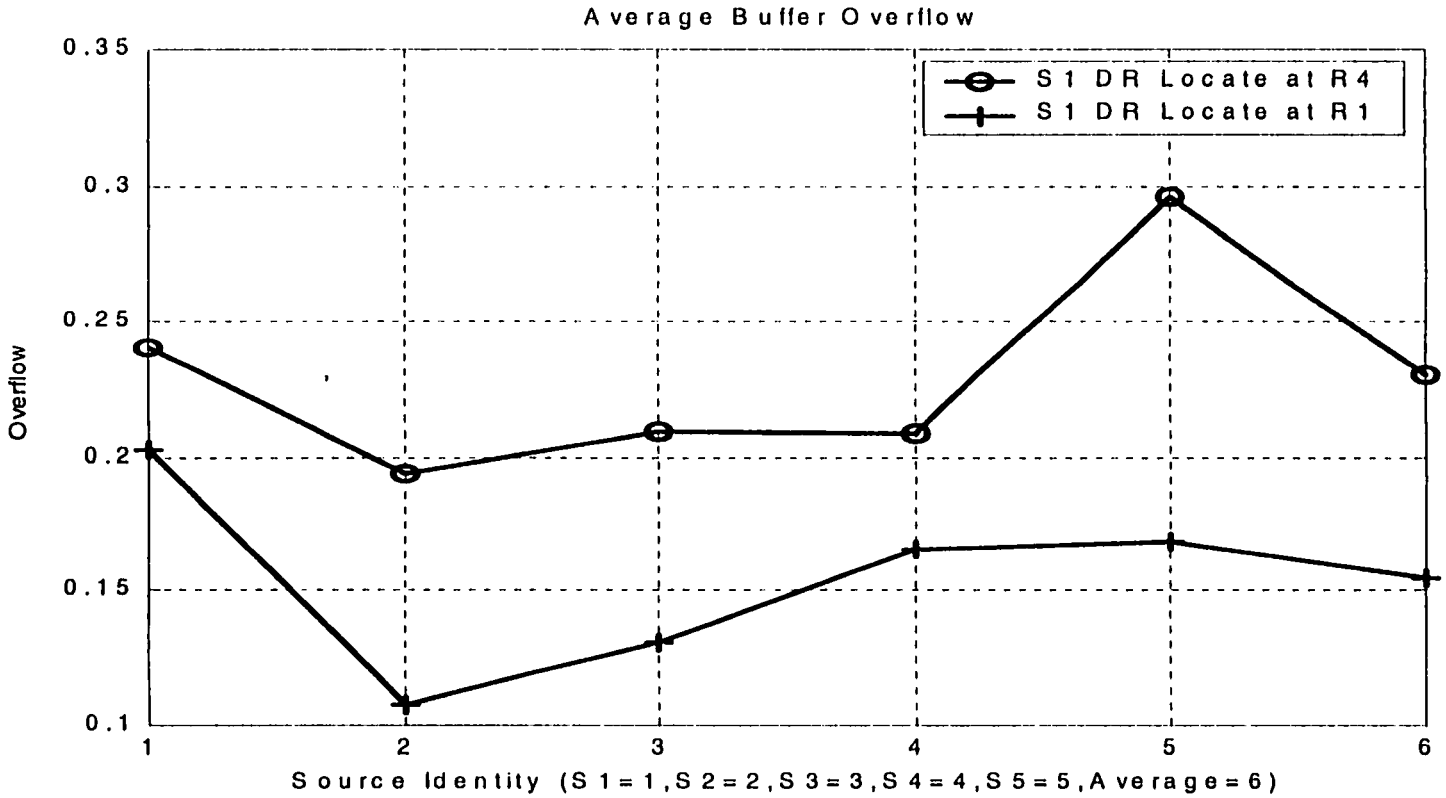


Figure 3-41: Average Buffer Overflow vs. S1 DR location changing
(ρ=1, Pc=0.99, Generation Rates=1000Packects/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

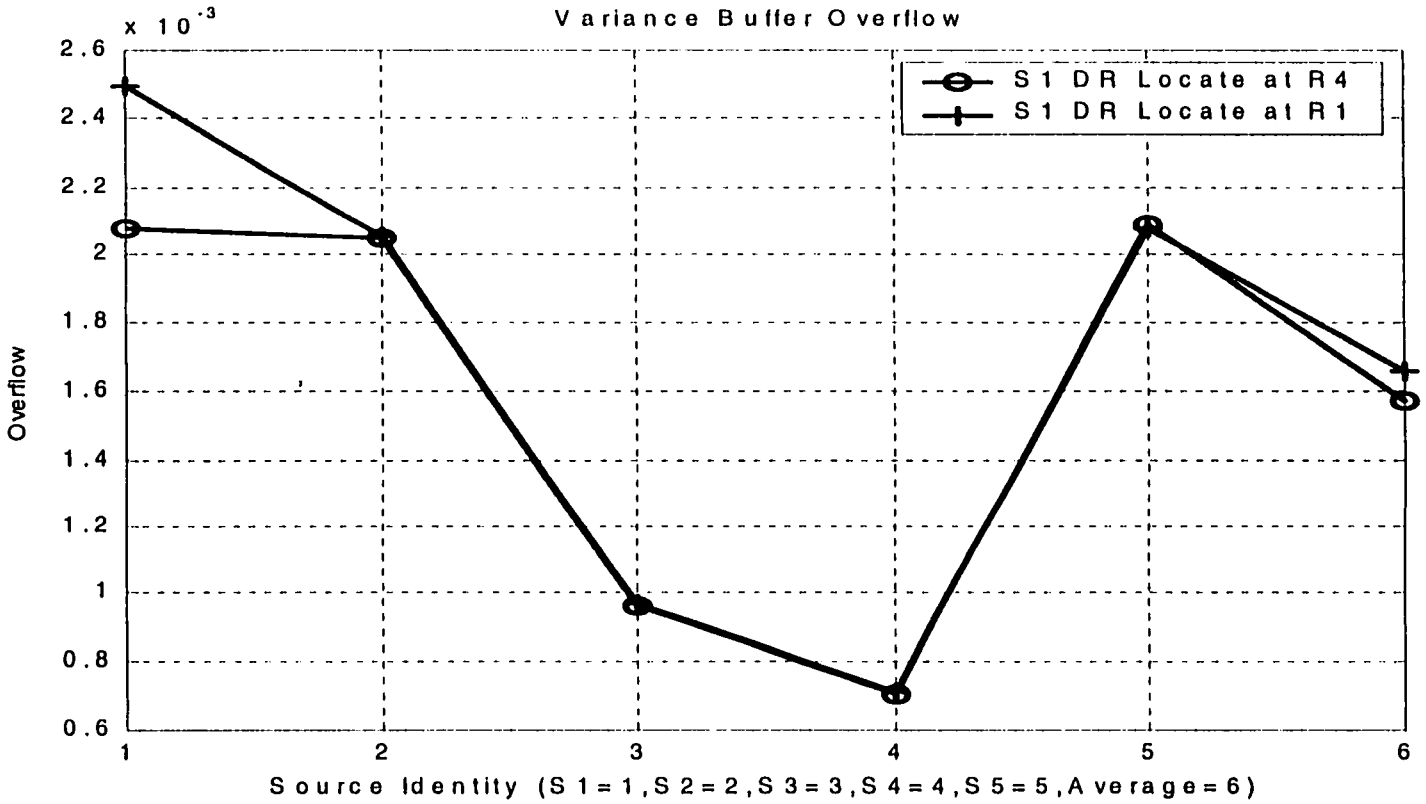


Figure 3-42: VarianceOverflow vs. S1 DR location changing
(ρ=1, Pc=0.99, Generation Rates=1000Packects/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

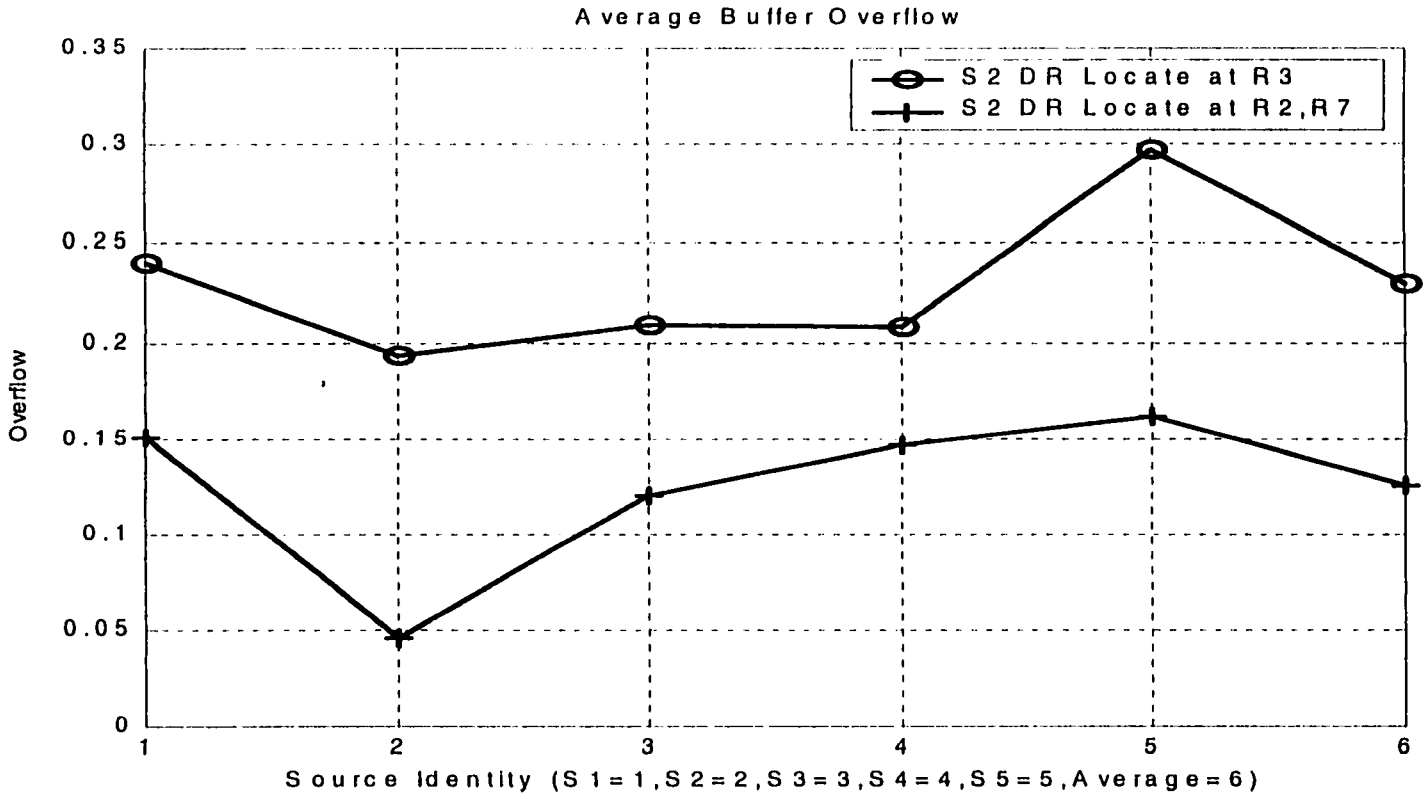


Figure 3-43: Average Buffer Overflow vs. S2DR location changing
(ρ=1, Pc=0.99, Generation Rates=1000Packeets/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

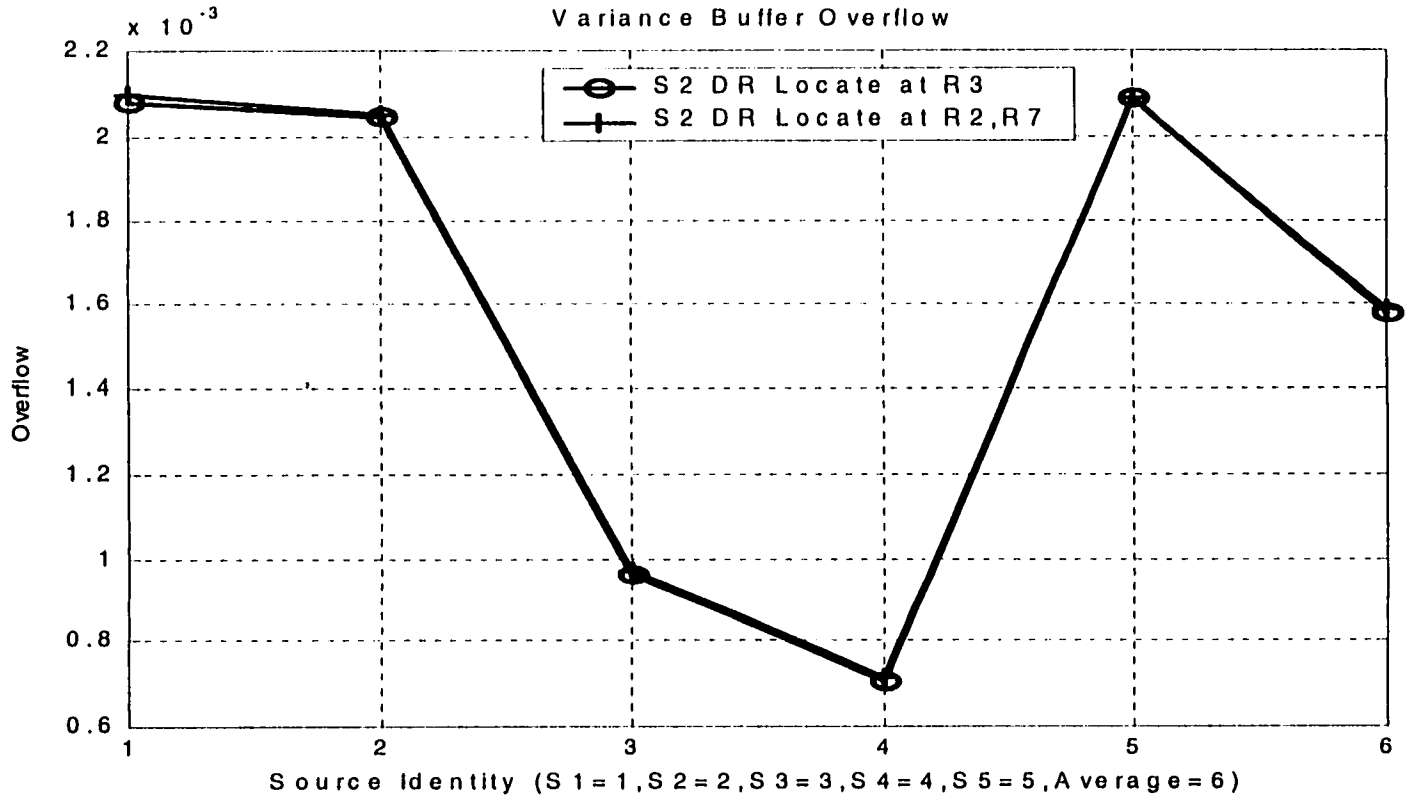


Figure 3-44: Variance verflow vs. S2DR location changing
(ρ=1, Pc=0.99, Generation Rates=1000Packects/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

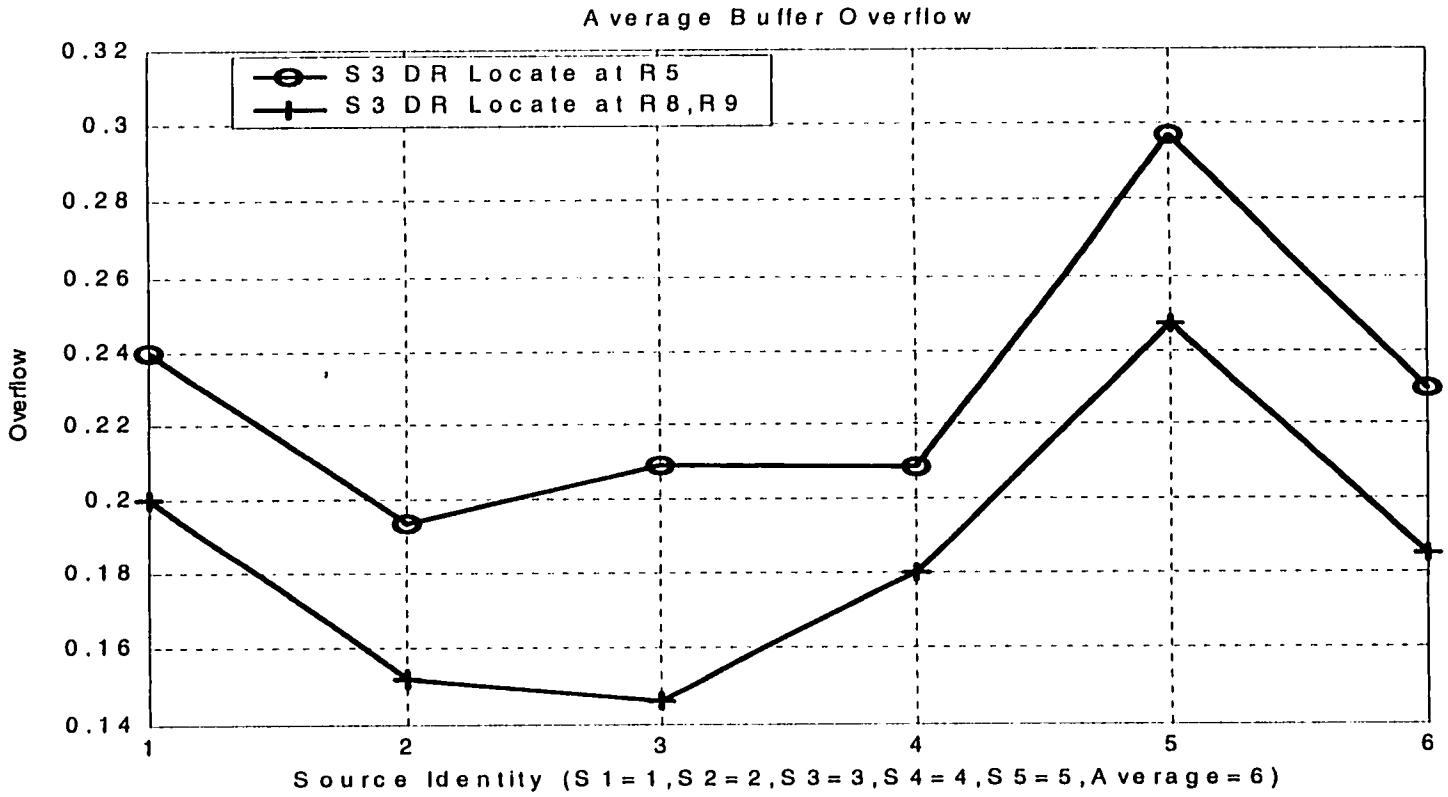


Figure 3-45: Average Buffer Overflow vs. S3DR location changing
(ρ=1, Pc=0.99, Generation Rates=1000Packects/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

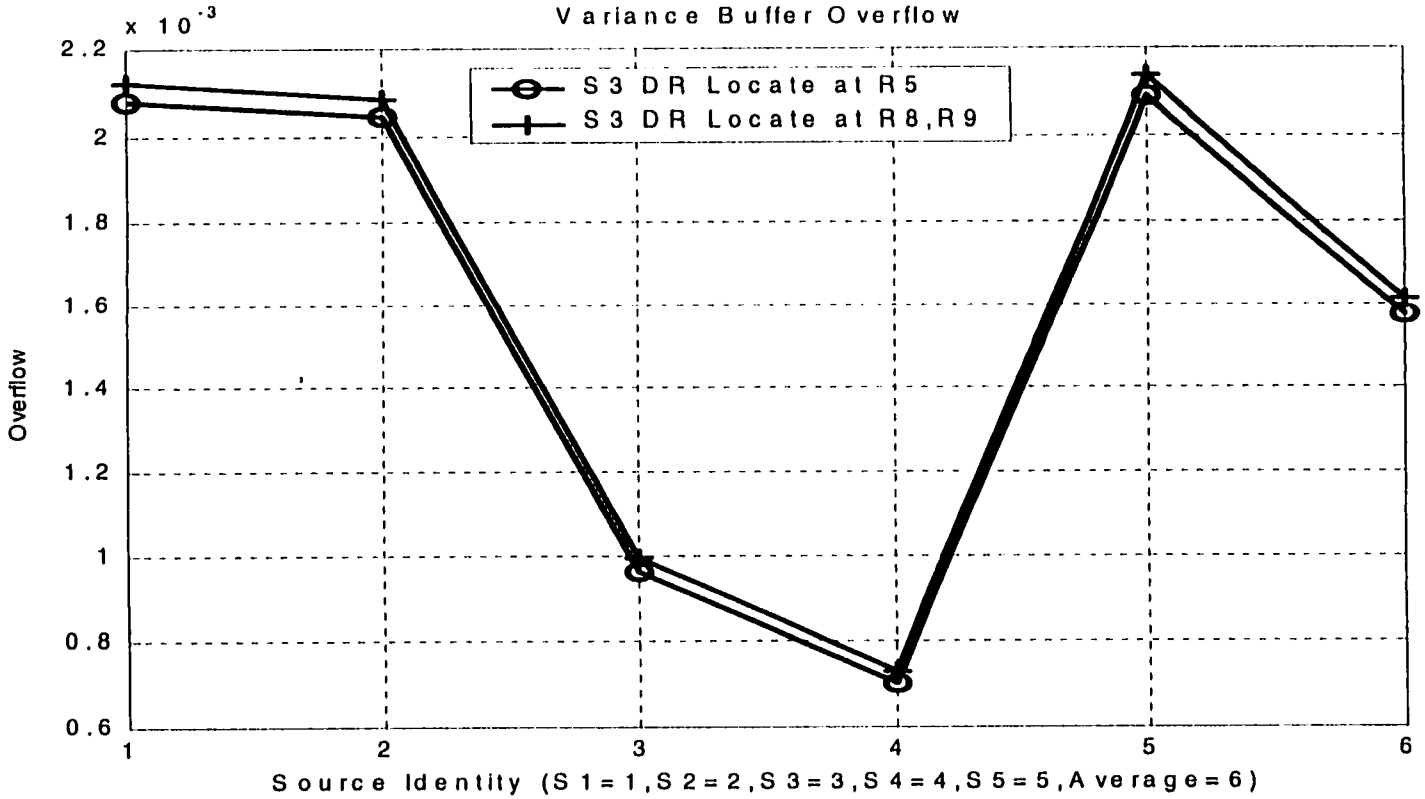


Figure 3-46: Variance Overflow vs. S3 R location changing
(ρ=1, Pc=0.99, Generation Rates=1000Packects/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

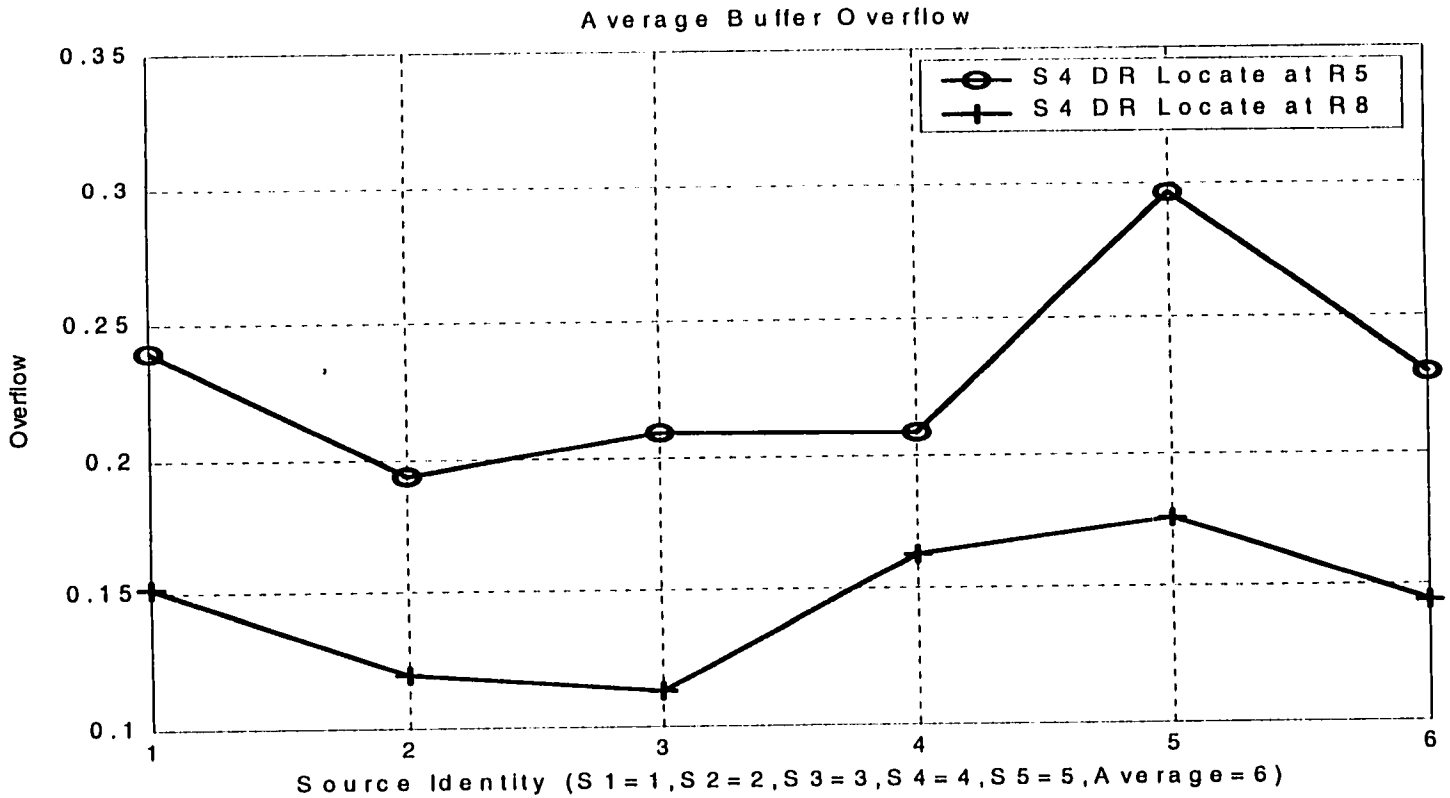


Figure 3-47: Average Buffer Overflow vs. S4 DR location changing
(ρ=1, Pc=0.99, Generation Rates=1000Packects/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

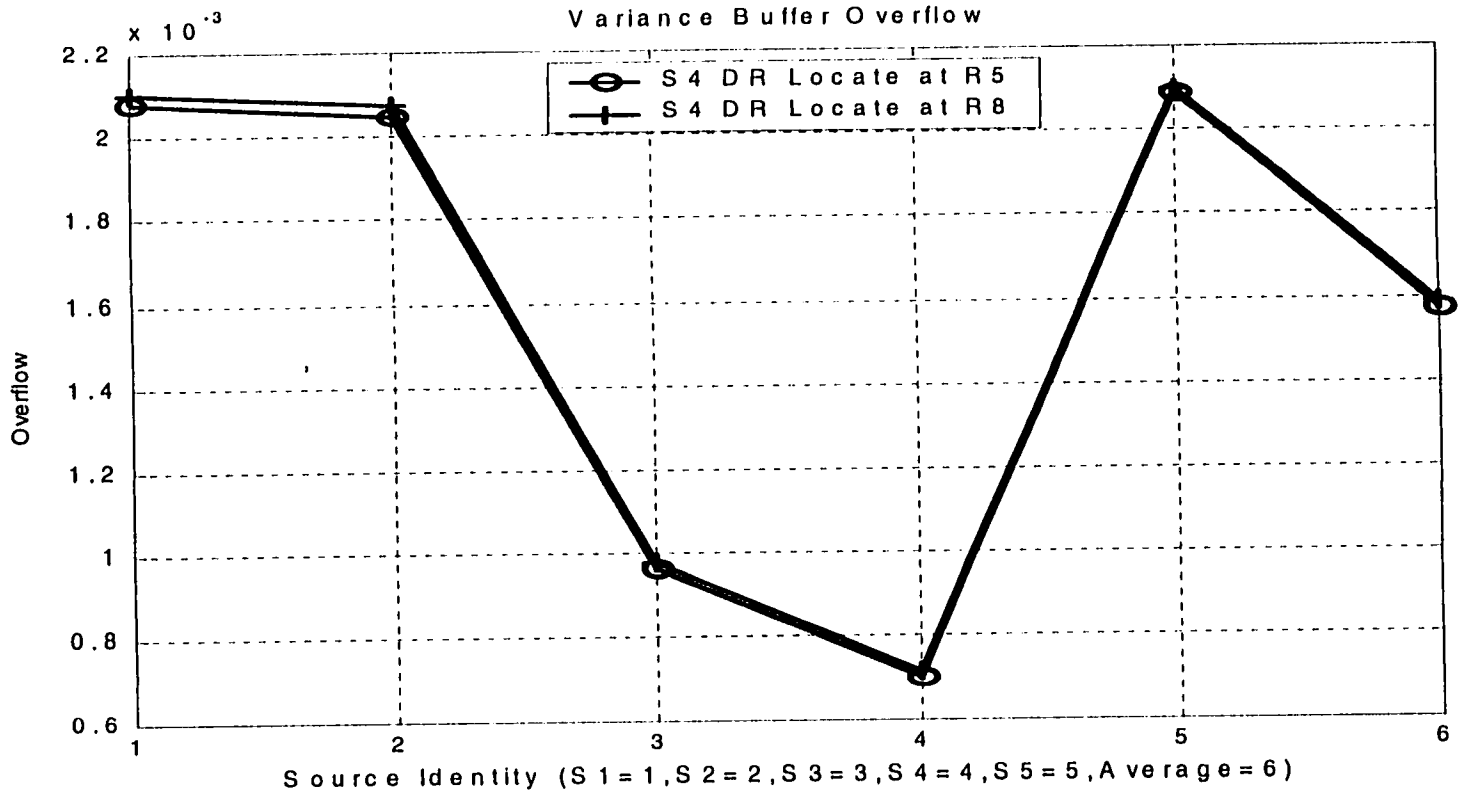


Figure 3-48: Variance Buffer Overflow vs. S4 DR location changing
(ρ=1, Pc=0.99, Generation Rates=1000Packects/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

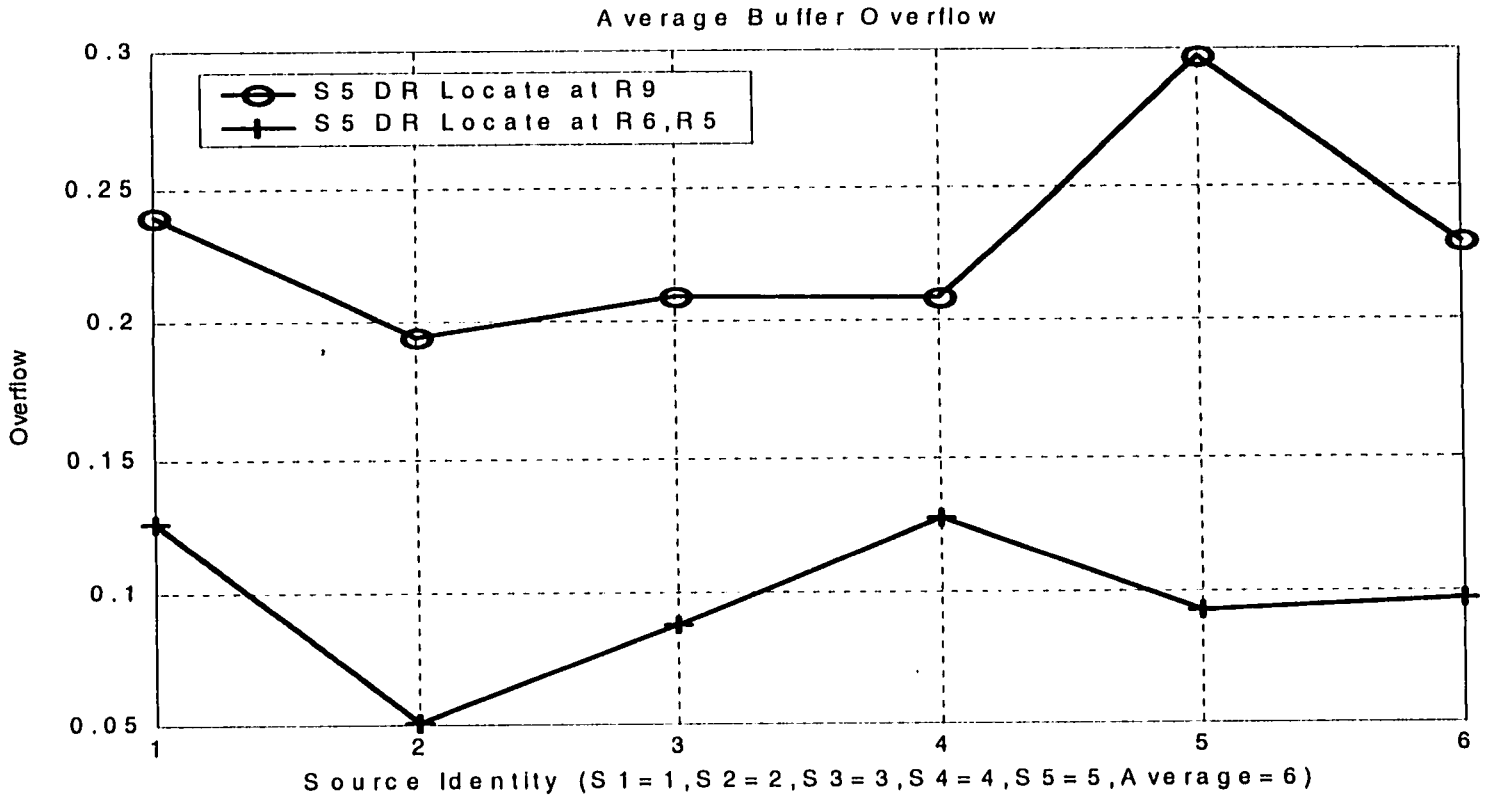


Figure 3- 49: Average Buffer Overflow vs. S5 DR location changing
(ρ=1, Pc=0.99, Generation Rates=1000Packects/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

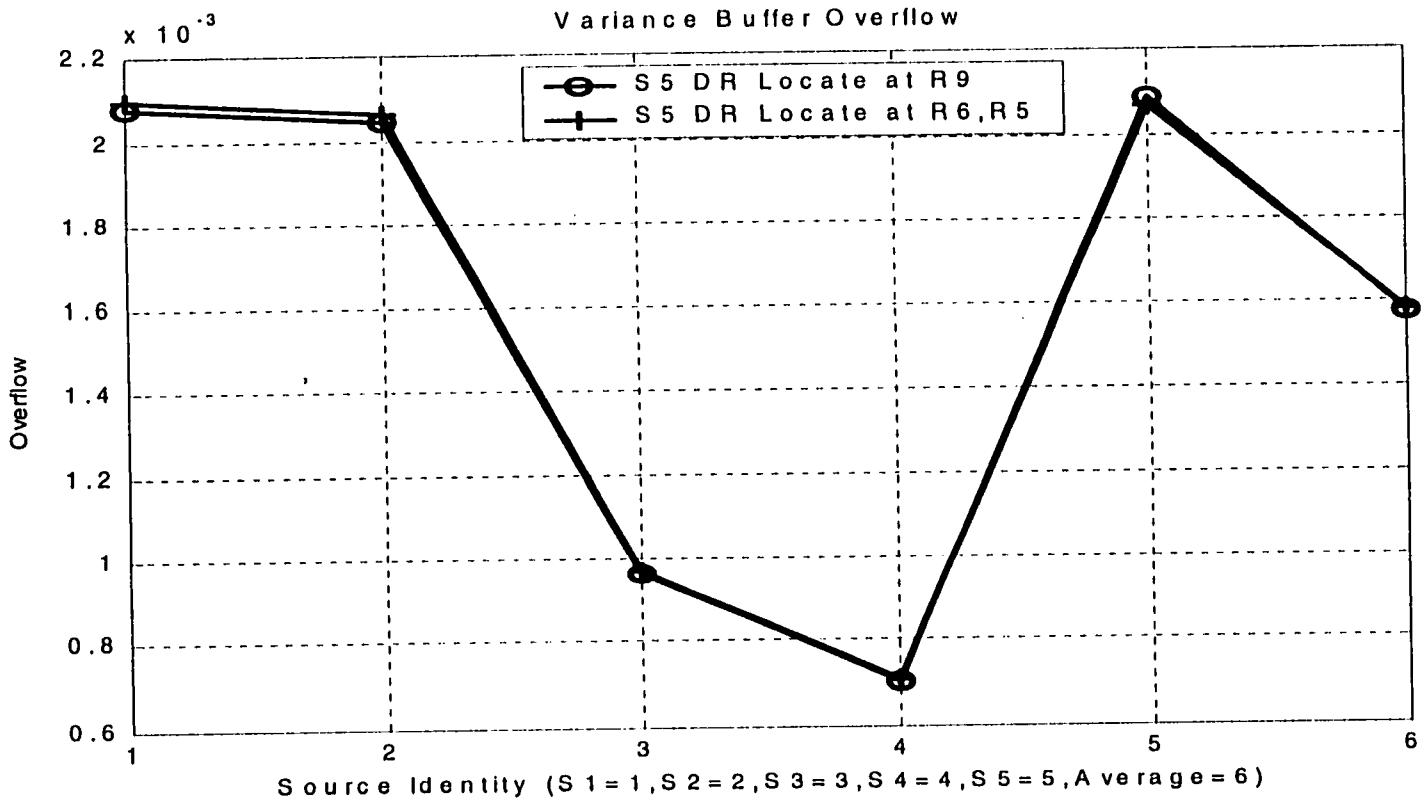


Figure 3-50 Variance Buffer Overflow vs. S5 R location changing
(ρ=1, Pc=0.99, Generation Rates=1000Packeets/Second, Buffer Size =50, TTL=62, Simulation Time=1e6)

CHATPER 4

SUMMARY AND FUTURE WORK

4.1 Summary

Here we summarize the research presented in this thesis. In Chapter 1, we introduced multicast's necessity and the basic concept behind it, Multicast addresses, trees, groups and protocols.

In Chapter 2, the concept of reliable multicasting is explored in depth. In addition, the chapter examines other major multicasting issues such as packet loss, error control types, error recovery types, implosion avoidance, and protocol organization.

In Chapter 3, a simulation study for the effects of DR location on QoS performance of reliable multicast is proposed. We give a detailed description of the scheme and its various parameters are presented. We begin with an analysis of the Average End-to-End Total Transfer Delay, followed by an evaluation of the Average Buffer Overflow and verification of End-to-End Throughput of the network. Results are shown with respect to various parameters trade off.

4.2 Conclusion

In this thesis, we propose a new loss control technology for reliable multicast. This technology could provide major reliable multicast functions such as:

--RS coding/ Selective Repeat ARQ/Interleaving

Hybrid FEC/ARQ Used Loss Control

--Hybrid Loss Recovery Used

--Delay Based Used for Avoidance of NAK Implosion

--Hierarchical Organization Used for Reduction in Recovery Latency Recovery Isolation

--Adaptation Multicast/ Unicast Retransmission Avoidance Retransmission Implosion

--Optimization on DR Location Choose

.From the simulation results, we can see the conclusions:

If we fix others parameters, and only change the source stream packets' generation rates, we see that as generation rates raises, average end-to-end total transfer delay increased, average end-to-end throughout decreases, and average buffer overflow increases.

If we fix other parameters, and only change the buffer sizes, we can see that the average end-to-end total transfer delay increases as buffer sizes are increased after low bound, average buffer overflow is decreased as buffer sizes are increased, and buffer size does not affect average end-to-end throughout very much.

Moreover, changing the position of the domain receiver (DR) is the most complex parameter. Spreading around DR location could decrease the average end-to-end total delay, except for one specific source streams close to the destination. For the end-to-end throughput, spreading around DR could decrease it, and the same for one specific source stream flow. For

the average buffer overflow, spreading around DR location could decrease it, and for one specific source stream close to the destination, which could decrease the delay for this, flow.

Tables show DR location changes:

Flow	DR location Close to Sender	DR location Close to Receiver	ATTD	VTTD
S1	R1	R4	R1 better	R4 better
S2	R3	R2 or R7	R2,R7 better	R3 better
S3	R5	R9 or R9	Same	R5 better
S4	R8	R5	Same	R5 better
S5	R9	R6 or R5	R6,R5 better	R9 better

Table 4-1: ATTD VTTD vs. DR Location change

Flow	DR location Close to Sender	DR location Close to Receiver	Average Throughput
S1	R1	R4	R4 better
S2	R3	R2 or R7	R3 better
S3	R5	R8 or R9	R5 better
S4	R8	R5	R5 better
S5	R9	R6 or R5	R9 better

Table 4-2: AT vs. DR Location change

Flow	DR location Close to Sender	DR location Close to Receiver	ABO	VBO
S1	R1	R4	R1 better	R1 better
S2	R3	R2 or R7	R2,R7 better	Same
S3	R5	R8 or R9	R8,R9 better	R5 better
S4	R8	R5	R8 better	Same
S5	R9	R6 or R5	R6.R5 better	Same

Table 4-3: ABO VBO vs. DR Location change

After we did our simulations, we compared it with other works. Harvard University H.T. Kung and S.Y. Wang did similar simulations [32]; they only used lost report and retransmission. The results are show in Figure 4-1; some parameters are list here such as NC: Number of clients in this experiment

RS: Server reply size in cells

BS: Buffer size, in cells, of the over-subscription input port of Switch 1

BPR: Background Peak Rate of the periodic square-wave traffic

BLP: Background Load Period, which is the length of the periodic square-wave cycle in cell cycles

BDR: Background Duty Ratio in each periodic square-wave cycle

Their average end to end throughput never exceeds 80 percent; whereas, as Figure 3-31 shows, my results always exceed 82 percent. These differences can be attributed to the advanced technology used in my simulation. Such as,

Hybrid FEC/ARQ Loss Control

Hybrid Loss Recovery Used

Delay Based Used for Avoidance of NAK Implosion

Hierarchical Organization Used for Reduction in Recovery Latency Recovery Isolation

Adaptation Multicast/ Unicast Retransmission Avoidance Retransmission Implosion

Optimization on DR Location Choose

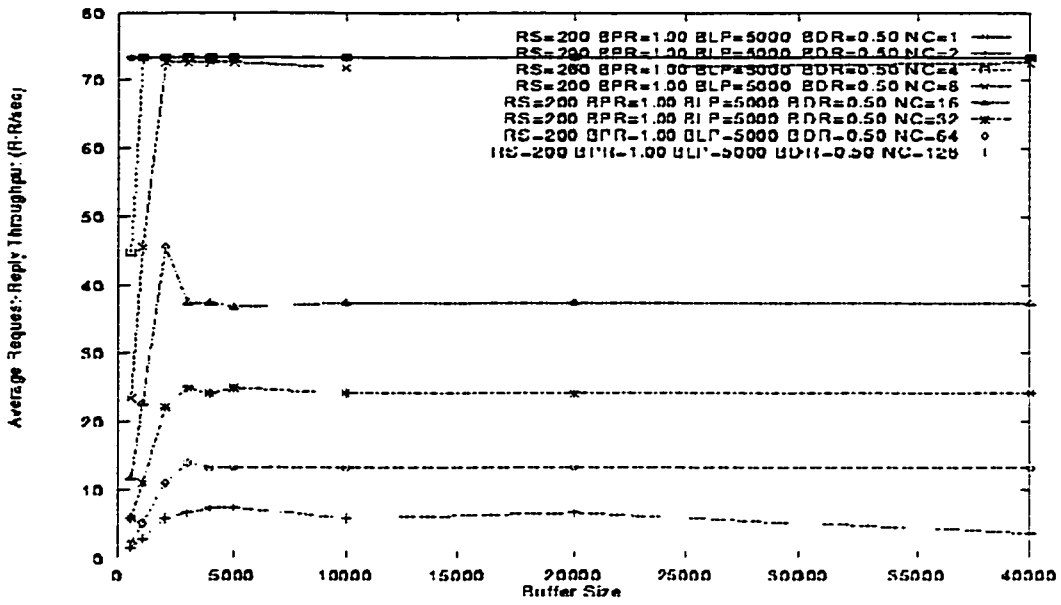


Figure 4-1: Throughput Results from Client-Server Performance on Flow-Controlled Networks

4.3 Suggestion for Future Work

The work described in this thesis is just a step towards optimization loss control technology for reliable multicast. There are still other features that were not explained in detail in this thesis. Also there remain some questions that need further study and investigation.

First, if we increase the deployment of multicasting, the issue of scalability becomes one of the most important aspects related to reliable multicasting. These aspects include the bandwidth requirements and limitations, the processing amount needed at senders and

receivers, and the complexity of the algorithms needed to manage a complex network structure. Thus the scalability of loss control should be further studied [14].

Second, it would be interesting to using Turbo Coding which could optimize the Reed Solomon code to have a has 2dB gain (at BER 10-7) instead of pure Reed Solomon code. Therefore, Turbo Coding embedded simulation also needs to be further studied [12].

At the end, in wired networks, changes in network topology are rare and link capacities are considered abundant. This is, however, not the case for mobile ad hoc networks. In a mobile ad hoc network, mobile nodes establish wireless connections among themselves on the fly, without any centralized coordinators. Otherwise, multi-hop connections are used in which packets are forwarded by one or more intermediate nodes until the destination node is reached. Because every mobile node can move, changes in network topology are frequent. Furthermore, the bandwidth of wireless links is an order of magnitude lower than that of wired links. So if we further study in an ad hoc environment, the loss technology must be highly adaptive to be able to cope with highly dynamic network conditions. In addition, they must be lightweight in terms of control overhead [28].

References

[1]J. Vicki and J. Marjory, "How IP Multicast Works - An IP Multicast Initiative White Paper", Web: www.ipmulticast.com, visited in Nov. 2001

[2] M. Hofmann, "A Generic Concept for Large-Scale Multicast", In B. Plattner, editor, Proc.International Zuerich Seminar, Volume 1044 of LNCS, pp.95-106, Springer Verlag, February 1996

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

[4] R. Braudes and S. Zabele, "Requirements for multicast protocols", Request for Comments(Informational) RFC 1458, Internet Engineering Task Force, May 1993

[5] S. Armstrong, A. Freier, and K. Marzullo, "Multicast Transport Protocol", Request for Comments (Informational) RFC 1301, Internet Engineering Task Force, February 1992

[6] E. Klemmer, "Subjective Evaluation of Transmission Delay in Telephone Conversations", Bell Systems Technical Journal, vol.46, pp.1141-1147, July 1967

[7] D. Towsley, J. Kurose, and S. Pingali, "A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols", IEEE Journal on Selected Areas in Communications, 15(3) pp.398-406, 1997

[8] S. Pejhan, M. Schwartz, and D. Anastassiou, "Error Control Using Retransmission Schemes in Multicast Transport Protocols for Real-Time Media", IEEE/ACM Transactions on Networking, 4(3) pp.413-427, June 1996

[9] S. B. Vicker, "Error Control Systems for digital communications and storage"

Prentice-Hall, 1995.

[10] J. Nonnenmacher, E. W. Biersack, and D. Towsley, " Parity-Based Loss Recovery for Reliable Multicast Transmission", IEEE/ACM Transactions on Networking, Vol. 6, No.4, pp.349-361, August 1998

[11] L. Rizzo ., and L. Vicisano , "Effective Erasure Codes for Reliable Computer Communication Protocols", ACM SIGCOMM Computer Communication Review, Vol.27, No.2, pp.24-36, Apr 1997

[12] J.H. Jeng , and T.K. Truong, "On Decoding of both Errors and Erasures of a Reed-Slolomon Code using an Inverse-Free Berlekamp-Massey Algorithm, IEEE Transactions on Communication, Vol. 47, No. 10, October 1999

[13] S. Lin and D. J. Costello, "Error Control Coding: Fundamentals and Applications", Prentice-Hall, 1983

[14] B. Rajagopalan, "Reliability and Scaling Issues in Multicast Communication", Proceedings of ACM SIGCOMM 92, pp.188-198, September 1992

[15] C. Papadopoulos, G. Parulkar, and G. Varghese, "An Error Control Scheme for Large-Scale multicast Applications", Proceedings IEEE INFOCOM'98 Conference on Computer Communications, vol.3 pp.1188-96, March 1998

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

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

[18] C. Diot, W. Dabbous, and C. J, "Multipoint Communication: A Survey of Protocols,

Functions and Mechanisms", IEEE Journal on Selected Areas in Communications, 15(3), pp.277-290, April 1997

[19] G. Carle, E. Biersack, "Survey on Error Recovery for IP-based Audio-Visual Multicast Applications", IEEE Network Mag., pp.24-36, Nov./Dec. 97

[20] J. P. Macker, "Reliable multicast transport and integrated erasure-based forward error correction", ProceedingsIEEE MILCOM, p.973-7, vol.2, November 1997

[21] M. Luby., G. Vicisano , L. Rizzo, L. Handley, M. Crowcroft, "The use of Forward Error Correction in Reliable Multicast", Internet draft draft-ietf-rmtinfo-fec-01.txt, October 2001

[22] L. Rizzo, "On the Feasibility of Software FEC", DEIT Tech Report, http://www.iet.unipi.it/~luigi/softfec.ps, Jan 1997

[23] R. Aiello, E. Pagani and G. P. Rossi, "Design of a Reliable Multicast Protocol", Proceedings of IEEE INFOCOM '93, pp.75-81, March 1992

[24] P. Danzig, "Flow Control for Limited Buffer Multicast", IEEE Transactions on Software Engineering,Vol. 20, No. 1, pp.1-12, Jan 1994

[25] J. Rosenberg and H. Schulzrinne, "An RTP Payload Format for Generic Forward Error Correction", Request for Comments 2733, Internet Engineering Task Force, Dec. 1999

[26] K. Obraczka "Multicast Transport Mechanisms: A Survey and Taxonomy", IEEE Communications Magazine, Vol. 36 No. 1, January 1998

[27] D. Villela, O. Duarte, "Improving scalability on reliable multicast communications", ELSEVIER Computer Communications Vol. 24, pp.563-574, 2001

[28] C.K Toh, "Ad hoc mobile wireless networks", PH PTR 2002

[29] A. Hanna, "Towards a Better Architecture for Reliable Multicast Protocols", M.Comp.Sc. Thesis, Department of Computer Science, Concordia University, Montreal, April 2000.

[30] B. Whetten and G. Taskale, "Reliable Multicast Transport Protocol II", IEEE Network, January/February 2000, vol. 14, no. 1, pp. 37–47.

[31] A. Tranenbaum, " Computer Networks ", Prentice Hall PTR 1996

[32] H.T. Kung and S.Y. Wang "Client-Server Performance on Flow-Controlled Networks" Harvard University 1996