An Integrated FEC/ARQ Scheme for Reliable Multicast over the Internet

Nasser Rajabieh Shayan

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

Montreal, Quebec, Canada

December 2002

Abstract

An Integrated FEC/ARQ Scheme for Reliable Multicast over the Internet

Nasser R. Shayan

The requirement of reliable communication protocols is that all the intended receivers of a message receive it unimpaired and complete. Automatic Repeat Request (ARQ) techniques are used in unicast protocols. The main advantage of this scheme is bandwidth conservation because no redundant packets are transmitted. ARQ based protocols do not scale well to multicast protocols with large groups of receivers. Effectiveness of retransmissions is reduced since losses tend to become independent and the advantage of conserving bandwidth does not hold. Forward Error Correction (FEC) is another method for loss recovery. Redundant packets are sent together with regular *data* packets from the sender to the receiver. Depending on how many packets are lost it may be possible to recover missing packets based on the received ones. However, FEC alone does not provide complete reliability. But combining FEC with ARQ results in significant improvements for reliable multicasting.

In this thesis we have implemented a code that integrates FEC with ARQ-based technique. We have performed the test of our proposed QOS scheme in the laboratory. The communication between two PCs was tested using this hybrid FEC/ARQ technique while network impairments such as packet drops were applied by a network impairment emulator. The performance of the system was studied. We have also simulated a multicast environment where our QOS techniques were used. The scheme was applied to a network consisting of different multicast flows. Various cases such as independent and shared losses for homogeneous receivers and also heterogeneous receivers were considered. Effect of using NAK implosion avoidance mechanism was also studied. Then behavior of the network considering queuing delays and losses due to buffer overflow was observed. In the end, the results were compared to ARQ based technique and improvements obtained by our proposed scheme were determined.

Dedicated to my wife for her support, encouragement and patience.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dr. A. K. Elhakeem for initiating the idea of my research and for his constant guidance through my work. I want to thank him for conducting our research meetings and his support during the entire course of this thesis. I would also like to thank École de Technologie Supérieure (ETS) for providing me with network laboratory facilities and instruments for performing practical tests of my thesis. In the end, I would like to thank my wife for her encouragement. Without her support and patience, I couldn't have finished this research.

Table of Contents

List of Tables

List of Figures

List of Abbreviations and Symbols

ACK	Acknowledgment
ARQ	Automatic repeat request
BNR	Number of bytes that are not recovered
BT	Number of transmitted bytes
CBT	Core based trees
CRC	Cyclic redundancy check
CW	Congestion window
DG	Number of data packets generated by sender
DL	Number of lost data packets
DR	Designated receiver (domain representative)
DVMRP	Distance vector multicast routing protocol
EBB	End of block burst
Eff	Efficiency
E(i)	Mean packet delay of flow i
E(i,j)	Mean packet delay from sender i to receiver j
FDDI	Fiber distributed data interface
FEC	Forward error correction
FIFO	First in first out (type of priority queues used in simulation)
GF	Galois field
IGMP	Internet group management protocol
K	Number of data bytes of RS code word
LAN	Local area network
LBRM	Log-based receiver-reliable multicast
LGMP	Local group multicast protocol
MDP	Multicast data protocol
MFTP	Multicast file transfer protocol
MOSPF	Multicast open shortest path first

MPEG	Moving picture experts group
mw	Memory allocation window
N	Number of total bytes of RS code word including data and parity
NAK	Negative acknowledgment
$N_p(i)$	Number of packets of flow i (from sender i)
NR	Percentage of retransmissions
$N_R(i)$	Number of receivers of flow I
P_b	Probability of background traffic at each iteration in a router
PIM-DM	Protocol-independent multicast-dense mode
PIM-SM	Protocol-independent multicast-parse-mode
PT	Maximum process time of one block of information
RBP	Reliable broadcast protocol
RINA	Receiver initiated with NAK avoidance (protocol)
RL	Residual loss
RM	Reliable multicast
RMP	Reliable multicast protocol
RMTP	Reliable multicast transport protocol
RS	Reed Solomon
Rt	Router
RT	Receive time of one block of information
RTCP	Real time control protocol
RTP	Real time transport protocol
RW	Number of retransmitted words
SCE	Single congestion emulation
Seq. No.	Sequence number
SIGIO	A Unix signal whose triggering event is "socket ready for I/O"
SMDS	Switched multi-megabit data service
SRM	Scalable reliable multicast
TCP	Transport control protocol
TMTP	Tree-based multicast transport protocol
TW	Total number of transmitted words excluding retransmitted ones

UDP	User data-gram protocol
WL	Window length, the time it takes to transmit packets of a block
W(n)	Receiver window for block n
XTP	Xpress transport protocol
$\bar{\Delta}$	Mean of Δ_t
Δ_{dec}	Reed Solomon decoding delay
Δ_t	End-to-end delay of data including delays in sender, network and receiver
$\Delta_k(i,j)$	End-to-end delay of packet number k from sender i to receiver j
Δ_{max}	Maximum of Δ_{dec} (when number of erasures equals maximum erasure correcting capability of decoder, i.e. N-K)
Δ_{min}	Minimum of Δ_{dec} (when number of erasures is 0)
λ	Total packet arriving rate to the router buffer
μ	Packet serving rate of router buffer
ρ	Traffic intensity of router buffer in packet arriving/packet served
$\sigma_{E_i}^2$	Variance of packet delay of flow i
σ_{Δ}^2	Variance of Δ_t

Chapter 1

Introduction

Many applications that use multicast communication have been developed in the recent years. These applications span an entire spectrum in terms of reliability and end-to-end latency requirements. Some of these applications, e.g. audio or videoconferencing tools tolerate segment losses with a relatively graceful degradation of performance. Others such as electronic whiteboards, electronic newspapers or distribution of software have instead more strict requirements and require reliable delivery of data. Thus they would greatly benefit from an increased reliability in the communication.

Reliable data transfer in computer communications is generally achieved by implementing reliability at different layers in the protocol stack, either at the link layer, or using end-to-end protocols like TCP at the transport layer, or directly in the application. The fundamental basis of achieving reliability is loss recovery. To recover from loss two well known techniques exist. First, automatic repeat request (ARQ) [1,2] that is generally used in unicast protocols: missing packets are retransmitted upon timeouts or explicit requests from the receiver. Possible retransmissions for several times will incur high latency. Also in multicast communication protocols ARQ might be highly inefficient because of uncorrelated losses at different groups of receivers. With the second technique, forward error correction (FEC) [2,22,23], sender prevents losses by

transmitting some amount of redundant information called parity along with the original data, which allow the reconstruction of missing data at the receiver without further interactions if the amount of lost original data is not more than received parity data. FEC by itself cannot provide full reliability. But when coupled with ARQ, FEC can be used to produce inherently scalable reliable multicast transport protocols. If FEC is introduced as a separate layer below ARQ layer, it causes reducing probability of packet loss and as a result reducing number of packet retransmissions and network bandwidth requirements. If integrated with ARQ in the same layer, FEC will have very high repair efficiency and, therefore substantially reduces network bandwidth requirements of an application requiring reliable multicast data transport.

In this thesis, we first implement a code based on our integrated FEC/ARQ scheme and test it in network laboratory of École de Technologie Supérieure (ETS), Montreal on two PCs communicating as a sender/receiver pair while a network impairment emulator PC applies errors and erasures to the transmitted packets. The scheme is receiver initiated i.e. the sender does not keep track of the state of receivers. The receivers send negative acknowledgements (NAKs) to the sender when the number of lost original packets are more than received parity packets. The performance of the experimental system is studied at various conditions such as flow rate and packet loss. It is seen that the code performs satisfactorily and reliable communication is achieved using the developed code. Then our scheme is expanded and applied to a hypothetical multicast network consisting of multicast flows with various topologies and different number of receivers. Simulation results show that the integrated FEC/ARQ scheme makes more efficient use of network resources than an approach solely based on ARQ even when losses are correlated and shared and also it is seen that adding FEC increases reliability and scalability.

The rest of the thesis is organized as follows. The remainder of this chapter gives an introduction to multicast communications and also a brief overview of FEC based reliable multicast protocols. Chapter 2 considers the test and performance of our implemented code on two PCs in ETS network laboratory. The focus of chapter 3 will be on multicast network performance in presence of independent or shared losses for homogeneous receivers as well as flows with heterogeneous receivers. The performance

is studied with and without NAK avoidance mechanisms. Also, behavior of our hypothetical multicast network considering queuing delay in routers while packet losses are caused by router buffer overflow (congestion), is studied. Conclusions are drawn in chapter 4.

1.1 Multipoint Communications

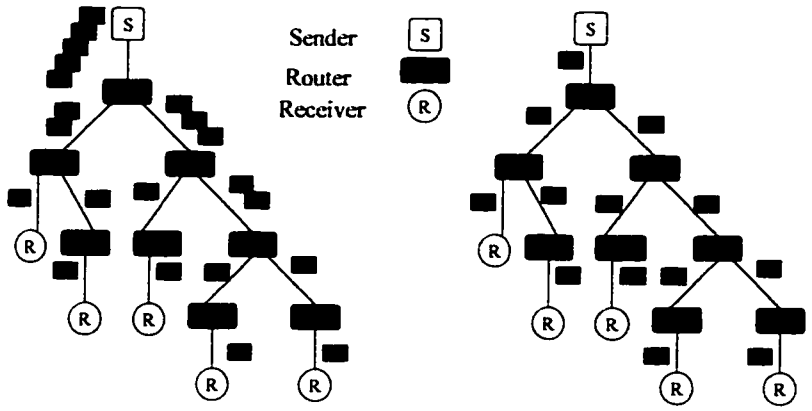
Many emerging Internet applications are one-to-many or many-to-many, where one or more sources are sending data to multiple receivers [1]. Transmission of data to multiple receivers can be done in three different ways as follows:

1. Unicast [2]. In multipoint unicasting, a source sends an individual copy of a message to each recipient (Figure 1.1). This technique is easy to implement but suffers from significant scalability restrictions if the group is large. The number of receivers is also limited by the sender's bandwidth.

2. Broadcast [2,3]. In a broadcast design, applications can send one copy of each packet and address it to a broadcast address. One significant feature of broadcast is to relieve the source from the task of duplicating any packet that is addressed to multiple receivers. If this technique is used the network must either stop broadcasts at local area network (LAN) boundary (as is frequently done to prevent broadcast storms) or send the broadcast everywhere which will reduce the available bandwidth to a large extent if only a small group actually needed to see the packets.

3. Multicast [3]. Multicasting falls between unicast and broadcast. Rather than send data to a single host (unicast) or to all hosts on a network (broadcast), with a multicast design, applications can send one copy of each packet and address it to the group of recipients (the host group) that explicitly want to receive it. Multicast is a receiver-based concept. Receivers join a particular multicast session group and traffic is delivered to all members of that group by the network infrastructure. The sender does not need to maintain a list of receivers. Only one copy of a multicast message will pass over any link in the network, and the copies of the message will be made only where paths diverge at a router (Figure 1.1). Thus IP

multicast yields many performance improvements and conserves bandwidth end-to-end.



a. With unicast, for every packet sender transmits as many copies as number of receivers.

b. With multicast, sender only transmits a single copy of a packet regardless of number of receivers.

Figure 1.1:Comparison of Multicast and Multiple Unicast

1.2 Multicast Applications

Many kinds of applications have already been developed to use multicast. Some of which cover multicast requirements of LANs and small networks. Some other were developed based on the needs of wide area multicasting made feasible by deployment of internet multicast backbone (Mbone) [2]. However, many of these can be divided into three broad categories based on reliability and latency [3,8]. On one end of the spectrum are interactive real time applications such as conferencing with stringent latency requirements. The typical latency requirement for this type is in the order of 100 ms. This type of applications can tolerate some loss because of the inherent redundancy in audio and video data. On the other hand of the spectrum are reliable multicast applications, such as document distribution or software distribution that require 100 % reliability.

Latency is not as critical for these applications as for the interactive real time applications. The third category of multicast applications falls between the two extremes in the sense that they have reliability requirements not as rigorous as reliable multicast applications, while their latency requirements are less stringent than the interactive real time applications. These are one-way non-interactive real time streaming applications such as streaming music or movie. Figure 1.2 shows categories of multicast applications based on their reliability and latency requirements. Some more examples of multicasting are web server replication, distribution of stock quotes and billing data, distance learning, distributed database applications, corporate messages and files and software updates sent to the staff.

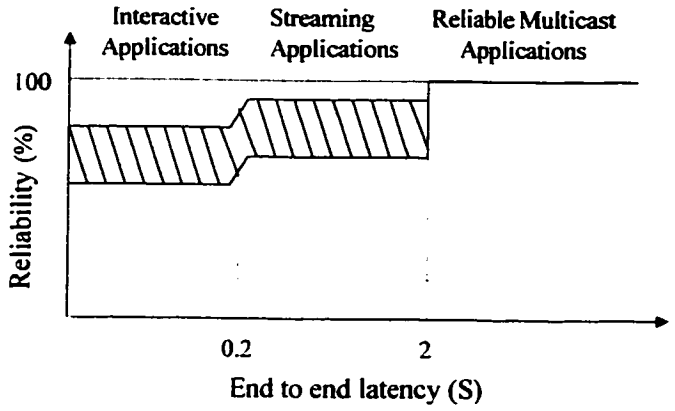


Figure 1.2: Categories of Multicast Applications Based on Reliability and Latency Requirements

1.3 IP Multicast and Data Link Layer Multicast [4,5]

Multicast can be implemented at both the data-link layer and the network layer. Ethernet and FDDI [2], for example, support unicast, multicast and broadcast addresses. An individual computer can listen to a unicast address, several multicast addresses, and the broadcast address. Token Ring also supports the concept of multicast addressing but uses a different technique. Token Rings have functional addresses that can be used to address groups of receivers.

If the scope of an application is limited to a single LAN, using a data-link layer multicast technique is sufficient. However, there are many multipoint applications that are not limited to a single LAN.

When a multipoint application is extended to an internet consisting of different networking technologies, such as Ethernet, Token Ring, FDDI, ATM, Frame Relay and SMDS, it is best to implement multicast at the network layer.

1.4 IP Multicast Delivery and Groups [4]

IP multicast as an extension to the IP network-layer protocol is described in RFC 1112 [5] as: " the transmission of an IP datagram to a 'host group', a set of one or more hosts identified by a single IP destination address. A multicast datagram is delivered to all members of its destination host group with the same 'best-efforts' reliability as regular unicast IP datagrams. The membership of a host group is dynamic; i.e.; hosts may join and leave group at any time. There is no restriction on the location or number of members in a host group. A host may be a member of more than one group at a time." In addition, at the application level, a single group address may have multiple data streams on different port numbers, on different sockets, in one or more applications. Multiple applications may share a single group address on a host.

1.5 Network layer requirements for multicasting

1.5.1 Addressing [2,6]

A network-layer address is needed to be used for communicating with a group of receivers rather than a single receiver. For this purpose, Class D addresses of IP address space that is reserved for multicast traffic is used (Figure 1.3). Classes A, B and C are used for unicast traffic. In addition, there must be a mechanism for mapping the IP address onto data-link layer multicast addresses where they exist.

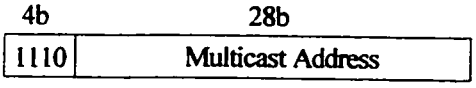


Figure 1.3: IP Class D Address for Multicast

1.5.2 Dynamic Registration [3,4]

There must be a mechanism for a host to communicate to the network that it is a member of a particular group. Without this ability, the network cannot know which sub-networks need to receive traffic for each group. RFC 1112 defines the Internet Group Membership Protocol (IGMP). IGMP specifies how the host should inform the network that it is a member of a particular multicast group. IGMP main ideas are as follows.

In order to determine if any hosts on a local subnet belong to a multicast group, one IGMP-capable router per subnet periodically broadcasts an IGMP Host-Membership Query message on its subnet. If a host on the subnet is a group member, a random timer is set to send one IGMP Host-Membership Report. When the timer expires, the host multicasts the Host-Membership Report to the group members (not broadcast to all hosts on the subnet). Now the router which subscribes to all groups, knows that there is a member on its subnet listening to a given group. Other group members on the subnet cancel their timers and do not transmit the scheduled IGMP Host-Membership Report. A host is not needed to explicitly inform the router when it leaves the group. When the router sends the next IGMP Host-Membership Query, if the router does not receive any IGMP Host-Membership Report it knows that there is no member left in its subnet for that specific group.

1.5.3 Multicast Routing [6]

The network must be able to build packet distribution trees that allow sources to send packets to all receivers. A primary goal of these packet distribution trees is to ensure that each packet exists only one time on any given network (that is, if there are multiple receivers on a given branch, there should only be one copy of the packets on that branch). Since the number of the receivers of a multicast session might be very large, the sender is not to know all the relevant addresses. Instead network routers should translate multicast addresses to host addresses. The basic principal in multicast routing is that routers interact with each other and exchange information about neighboring routers. For each physical network one router is selected (via IGMP) so that effort duplication is avoided. Selected routers make a spanning tree (Figure 1.4) that covers all members of a multicast

group. There is maximum one connection between every pair of routers and the tree is loop free. If each router knows which of its links are in the multicast tree, it can generate only the required copies of incoming multicast message and send them to its outgoing branches. Messages are only replicated when the tree branches, so minimum copies of them are transmitted through the network. The spanning tree must be dynamically updated in order to consider joining or leaving of members of a group at any time. Branches with no group members are discarded (pruned). Each router joins the multicast tree based on the network layer source address of the multicast packet and prunes it based on the packet network layer destination (multicast) address. How routers interact with each other and which spanning tree algorithm is used, depends on the goals of routing protocol. Various routing protocols have been developed with different features and objectives as given in the next section.

1.5.4 Multicast Routing Protocols [7]

Multicast routing protocols can be categorized into two types depending on the distribution of members of multicast groups. For the first type, it is assumed that the multicast group members are densely distributed throughout the network and bandwidth is almost unlimited. These dense mode protocols periodically flood the network to setup and maintain multicast trees. Distance Vector Multicast Routing Protocol (DVMRP), Multicast Open Shortest Path First (MOSPF), and Protocol-Independent Multicast-Dense Mode (PIM-DM) [7] are dense-mode routing protocols. For the second type, group members are sparsely distributed and the network bandwidth is not necessarily widely available. Flooding the network in this case will waste network bandwidth, so more selective techniques have to be used by sparse-mode multicast protocols to setup and maintain multicast trees. Core based Trees (CBT) [6,7] and Protocol-Independent Multicast-Sparse Mode (PIM-SM) [7] are sparse-mode routing protocols.

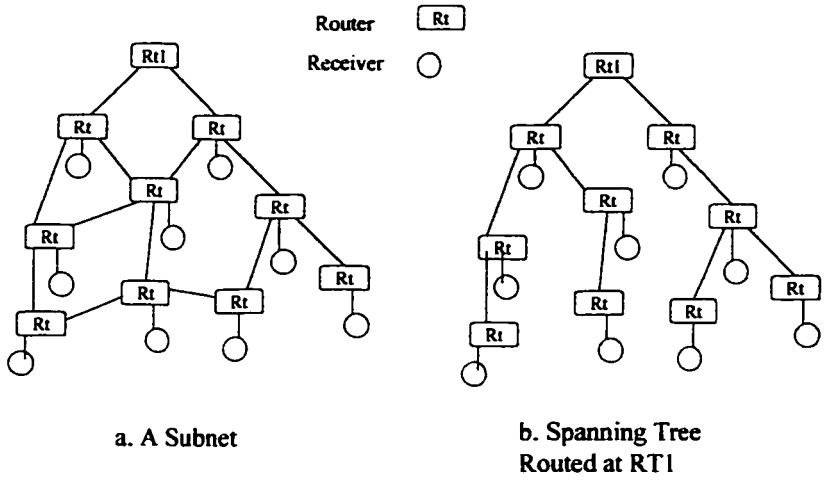


Figure 1.4: Spanning Trees

1.6 Transport Layer Multicast Protocols

End-to-end use of multicasting requires protocols for the transport of data as well as application-level protocols. Generally there are two types of these protocols. First groups are protocols developed for real-time multimedia data, like, audio, video or simultaneous data delivery such as Real-Time Transport Protocol (RTP) and its associated protocol, Real Time Control Protocol (RTCP). Second group are Reliable Multicast Protocols that provide reliable data delivery to multicast groups. These will be discussed in more detail in next section.

1.6.1 Reliable Multicast Protocols [8]

General data delivery using generic protocols is accomplished via IP and UDP, leading to unreliable delivery. This is unsuitable for many applications that can use multicasting. Therefore, other protocols have to be designed to add reliable delivery to multicasting. Reliable multicasting brings with it a series of problems such as, (a) the possibility that feedback from receivers might overwhelm a source (a feedback implosion, Figure 1.5); (b) how lost packets are recovered; and (c) the different requirements that different

multicasting applications can impose. Many reliable multicast protocols have been evolved out of necessity for solving specific problems and satisfying the needs of different applications. Therefore, their design criteria have been different. In spite of these differences several unique features can be used to group most of these apparently different protocols as follows:

a. Unicast-emulation protocols

Reliable multicast is implemented as an abstract form of unicast by these protocols. Both flow-control and error-recovery are sender based. Receivers send their retransmission request (NAK) to the sender and the sender retransmits them. Flow control is based on the feedback from the receivers. Two important protocols in this group are Xpress Transport Protocol (XTP) [9] and Single Congestion Emulation (SCE) [10]. SCE uses all features of TCP, i.e. flow/congestion control, error recovery and connection establishment.

b. Cycle-based Protocols

A file is divided into a sequence of fixed size packets such that each packet has a unique identifier (e.g. a sequence number) and the whole file is transmitted to all the receivers of a multicast group. After a transmission cycle, retransmission cycles start in which the receivers send the list of missing packets to the sender and the receiver retransmits them. IMM protocol [1], MFTP [12] and RMTP [13] are examples of these protocols. However, details of how flow/congestion control or how error control is done are different. IMM protocol is modified and completed by using forward error correction and NAK suppression using a timer based mechanism to avoid NAK implosion. This protocol is called MDP [14].

c. Tree based protocols

These protocols use divide and conquer policy. The recipients are grouped into local regions or domains with a representative called Designated Receiver (DR). The local regions are organized in a hierarchical tree structure. The receivers in a

local region send their requests for missing packets to their DR and the DR sends the missing packets to the receivers in the local region. If the DR does not have the requested packets, it requests them from the higher level DR until it reaches the sender. This approach reduces end-to-end latency by using local regions and solves the problem of NAK implosion by design, and therefore scales well in a wide area network where members are widely distributed. The protocols belonging to this group are RMTP, TMTP [15], LBRM [16], LGMP [17] and LORAX [18]. However these protocols differ in some details. Unlike other protocols in this category, LORAX is a many to many reliable multicast protocol.

d. Group communication protocols

These category of protocols provide different ordering and delivery semantics to the applications. Most of them provide total ordering across multiple senders, while some of them even provide k-resilient or majority-resilient delivery semantics. Reliable Broadcast Protocol (RBP) that belongs to this category was designed for local area broadcast networks. RMP [19] extended RBP for using it in a wide area network. RMP uses a TCP like flow/congestion control mechanism which relies on slow start, additive increase and multiplicative decrease of the sender's window, and also provides a group membership protocol to keep track of dynamically changing set of members. Scalable Reliable Multicast (SRM) [20] fits into this category of protocols for sharing documents among group members. However it does not provide order in delivery. This improves latency when the reliable delivery is much more important than the order of delivery. SRM is a NAK based protocol in which the receivers multicast NAKs to all group members and a timer-based mechanism is used to reduce the number of retransmissions sent for the same lost packet.

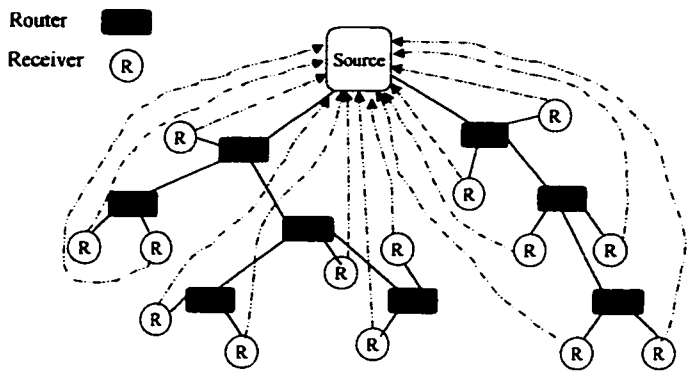


Figure 1.5: A Feedback Implosion

There is a classification for reliable multicast protocols based on how the different protocols adjust their memory allocation window (MW) and the congestion control window (CW). The memory allocation window is related to releasing of buffer associated with a block of data while the congestion window is associated with the rate of transmissions or retransmissions.

1. ***Sender-initiated protocols***

   In the sender-initiated protocols (Figure 1.6.a), the sender maintains the state of all receivers. Receivers send acknowledgements (ACKs) directly to the sender and after receiving ACKs from all the receivers the sender can decide to advance the memory allocation window (MW). These protocols suffer from ACK-implosion problem i.e. the sender has to process a great number of ACKs and this increases sender's processing load. XTP is an example of a sender-initiated protocol.

2. ***Receiver-initiated protocols***

   Receiver-initiated Protocols do not keep track of the receivers (Figure 1.6.b). Receivers send negative acknowledgments (NAKs) to the sender if they miss a packet. The sender retransmits the missing packet. In this scheme, if each receiver

sends a NAK to the sender, there will be a NAK implosion at the sender. There is a modification to this protocol that avoids NAK implosion. If a receiver detects a missing packet it schedules a timer. If the receiver hears a NAK before its timer expires, it cancels its timer. Otherwise it multicasts a NAK after expiration of its timer. This modified version of receiver-initiated protocol is named receiver-initiated with NAK avoidance (RINA) protocol. There is no mechanism for memory allocation window (MW) in these protocols since the sender has no idea when it is suitable to de-allocate the buffer space. However, the pacing of the receivers and retransmission mechanism i.e. adjustment of congestion window (CW) is done in a scalable and efficient manner. SRM is one of the protocols belonging to this category. Scalability [21] for SRM is achieved by NAK avoidance and application level framing.

3. *Tree-Based protocols*

In these protocols, the receivers are organized in a tree such that the sender is at the root of the tree, the receivers are at the leaves and the domain representatives are at the intermediate points of the tree. Domain representatives (DRs) represent a group of receivers or a domain and are also organized in a hierarchical manner. Sender is the highest level DR. In these protocols, the receivers send status messages (ACK+ bitmap or ACK+ NAK) to the corresponding DR. The sender/DR moves the congestion window (cw) based on the ACKs and retransmits missing packets based on the NAKs. RMTP, LGMP, TMTP belong to this category. Tree based protocols are optimized by using NAK avoidance scheme of the RINA protocols. These protocols are named Tree-NAPP because they combine NACK avoidance and periodic polling with the basic tree-based organization of the receivers. TMTP belongs to this group of tree-based protocols (Figure 1.6.c).

4. Ring-Based protocols

Ring-Based protocols (Figure 1.6.d) arrange the receivers in the form of a ring (on the contrary to the sender-initiated and RINA protocols which assume a flat organization of receivers). One of the receivers at any instant of time is designated a token site. The token site is responsible for sending ACKs back to the sender. The token site retransmits the missing packets. The token is passed to the next receiver in the ring when the next receiver has received all the packets that the current token site has received. Once the token is passed, the token site can move the memory allocation window (mw). Adjustment of the congestion window (cw) is done either by token site or by the sender. RMP and TRP are examples of this group of protocols.

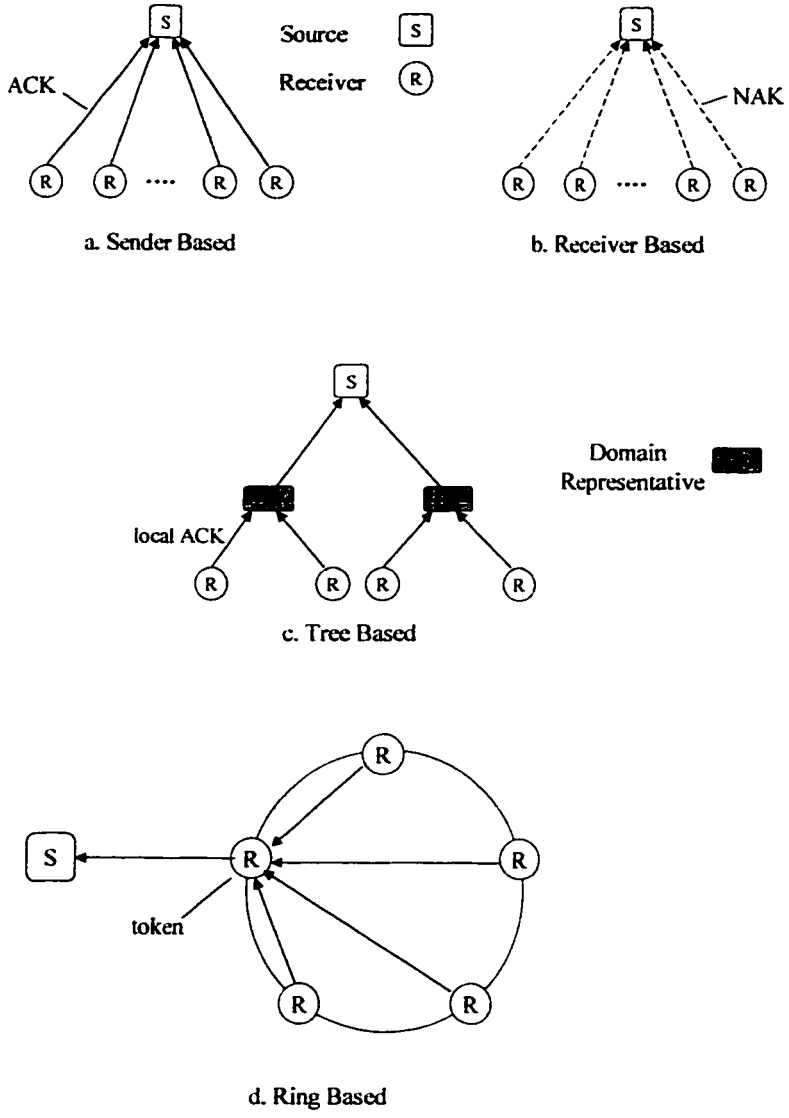


Figure 1.6: Different Classes of Reliable Multicast Protocols

1.7 FEC Based Reliable Multicast Protocols

Reliable data transfers among communicating parties is usually achieved by implementing reliability at different levels in the protocol stack, either on a link-by-link basis (e.g. at the link layer), or using end-to-end protocols at the transport layer (such as TCP), or directly in the application. The fundamental basis of reliable multicast protocols is loss recovery. There are traditionally two different ways of recovering missing packets:

1. Automatic repeat requests (ARQ)
2. Forward error correction (FEC)

1.7.1 Automatic Repeat Request (ARQ)

These techniques are generally used in unicast protocols. Missing packets are retransmitted upon timeouts or explicit requests from the receiver. When the bandwidth-delay product approaches the sender's window, ARQ might result in reduced throughput. The main advantage of this scheme is bandwidth conservation because no redundant packets are transmitted. However, the disadvantage of retransmission-based recovery mechanism is the increase in end-to-end latency because at least a round-trip elapses between the time a receiver sends a retransmission request and actually receives the transmitted packet from the sender. In case of multicast even the advantage of conserving bandwidth may not hold if the retransmission is also multicast regardless of the number of receivers that request the missing packet. And it might also be inefficient because of uncorrelated losses at different (groups of) receivers [22].

1.7.2 Forward Error Correction (FEC)

These techniques are generally based on the use of error detection and correction codes. These codes have been studied for a long time and are widely used in many fields of information processing, particularly in telecommunications systems. In the context of computer communications, error detection is generally provided by the lower protocol layers that use checksums (for example Cyclic Redundancy Checks) to discard the corrupted packets. Error correcting codes are also used in special cases, e.g. in modems,

wireless or otherwise noisy links, in order to make the residual error rate comparable to that of dedicated, wired connections. After such link layer processing, the upper protocol layers have mainly to deal with erasures, i.e. missing packets in a stream. Erasures originate from uncorrectable errors at the link layer (those are not frequent with properly designed and working hardware), or, more frequently, from congestion in the network which causes otherwise valid packets to be dropped due to lack of buffers.

Despite an increased need, and a general consensus on their usefulness there are very few Internet protocols that use FEC techniques [23]. This is possibly due to the existence of a gap between telecommunications world, where FEC techniques have been first studied and developed, and the computer communications world. In the former, the interest is focused on error correcting codes, operating on relatively short strings of bits and implemented on dedicated hardware; in the latter, erasure codes are needed, which must be able to operate on packet sized data objects, and need to be implemented efficiently in software using general purpose processors.

1.7.3 FEC Based Loss Recovery

In forward error correction-based recovery, redundant packets (also called *parity* packets) are sent together with regular data packets from the sender to the receiver. Depending on how many redundant packets are sent and how many packets are lost at the receiver, it is possible to recover missing packets on the received packets. Thus FEC-based approach reduces the end-to-end latency at the cost of additional bandwidth. Reed Solomon correcting code is chosen as a possible code because of its excellent capability for correction of both errors and erasures [24]. The details of code and preparing data and parity packets will be given in chapter 3. Here some general information is given. Suppose that there are K data packets denoted by $\{d_1, d_2, \ldots, d_K\}$. Using RS encoder and some processing N-K parity packets $\{p_1, p_2, \ldots, p_{N-K}\}$ are generated. Assuming that the data packets are transmitted in the order $d_1, d_2, \ldots, d_K, p_1, p_2, \ldots, p_{N-K}$, the following three scenarios can occur:

1. None of the K data packets are lost. Then no decoding is necessary.
2. L <= (N-K) packets are lost. This means the decoder can recover the missing packets.

3. L >= (N-K) packets are lost. In this case the lost packets can not be recovered unless additional q packets are transmitted, where Q = L-(N-K).

1.7.4 Combining FEC and ARQ

There are a number of different ways that FEC can be introduced to a reliable multicast protocol stack. The simplest approach is to add a layer responsible for FEC between the network layer and the reliable multicast layer, such that the reliable multicast layer sees a more reliable network. The second approach is to integrate FEC with reliable multicast and place them into a single layer. Layered FEC reduces packet loss probability and as a result reduces the number of packet retransmissions and network bandwidth requirements [25, 26]. FEC will have very high repair efficiency if it is integrated with ARQ in the same layer, and, therefore substantially reduces network bandwidth requirements of an application requiring reliable multicast data transport [26, 27].

In this thesis we have used integrated FEC approach as a basis for a new reliable multicast protocol. In chapter 3 the details of our implementation are given and the results of experiments performed on a single link between a sender and a receiver are discussed. In chapter 4 our integrated FEC technique is expanded to multiple receivers and multicast flows and by simulating a variety of different situations, performance of this technique is investigated. Protocol stack for our reliable multicast scheme integrated with FEC is shown in Figure 1.7.

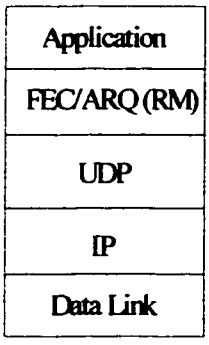


Figure 1.7: Integrated FEC/Reliable Multicast (RM) Protocol Stack

Chapter 2

Implementation and Testing of Integrated FEC/ARQ

In this chapter we introduce our technique where we integrate Forward Error Correction and Automatic Repeat Request as a receiver initiated [28] scheme to recover losses encountered in the Internet. In sections 2.1 to 2.3 details of FEC, sender/receiver algorithms, packet formats and interleaving are given. For FEC we have used a Reed-Solomon code because of its good error and erasure correcting capability. Details of preparing encoded words such as encoding, preparing parity information and adding CRC bytes are given and then interleaving the encoded words and adding 4 of such interleaved words to make a UDP datagram (packet) for transmission, are explained. NAK packet formats containing sequence number of required words are also given. In section 2.4 sending and receiving processes and programs are discussed. First, UDP socket programming done on Linux/Unix is described. Unix SIGIO signal that we have used for more efficient performance of the sender and receiver codes are explained. The receiving process provides information about checking the socket for new received words, checking for erasures, de-interleaving words, decoding and sending NAKs. Flowchart of the receiver code is also drawn in this section. Then window mechanism for our

FEC/ARQ scheme is described. In the last part of the section the sending process consisting of packet preparation and transmission, checking socket for NAK, storing the requests in a queue and serving them based on FIFO is considered. An interesting feature of our scheme stated in this chapter is the reduction of no of NAKs compared to general ARQ schemes without FEC. In error recovery schemes based on ARQ, NAK is sent as soon as a single packet is found to be missing but in our FEC/ARQ scheme only one NAK is sent for a whole block of packets. Focus of section 2.5 is on experimental details. First the test bed used in ETS network laboratory in Montreal is described. It consists of the two sending and receiving PCs that are interconnected through a third PC as a network impairment emulator. Our implemented codes for the new FEC/ARQ scheme are installed and run on the sender/receiver PCs. Various performance parameters such as mean delay, delay variance, no of retransmissions, no of NAKs, no of packets recovered by decoding and etc. are measured and their graphs versus transmission rate (throughput) and random packet loss are drawn. In the end of the chapter conclusions are drawn. It should be mentioned that in the thesis, flow rates are for packets containing data, parity and other redundant fields unless it is stated to be the actual data flow rate. Actual flow rates for data are less. For instance, for a value of 2Mbps the actual flow rate for data is 2*(220/256)=1.7 Mbps.

2.1 Reed-Solomon Code

In our implementation, a Reed-Solomon (RS) code by Phil Karn [29] is used. Complete details of these can be found in coding theory textbooks [30]. Here is a brief summary of the code: We have used a (N, K) RS systematic code over Galois Field or GF(2^M) where M is the code symbol size in bits, K is the number of data symbols per block, K<N and N is the block size in symbols, which equals (2^M -1). Since the Reed-Solomon Code is non-binary each RS "symbol" is actually a group of M bits and just one bit error anywhere in a given symbol spoils the whole symbol. That's why RS codes are often called "burst-error-correcting" codes [31]. In our implementation we have chosen values of 255, 223 and 8 for N, K and M respectively. The error-correcting capability of a Reed-Solomon code depends on N-K, the number of parity symbols in the block. In the pure error-correcting mode (no erasures indicated by the calling function), the decoder can correct

up to (N-K)/2 symbol errors per block and no more. The decoder can correct more errors if the locations of errors are known (i.e. erasures). For the case where the code is non-binary knowing error locations by itself is not enough to correct them. In the general case when there are both errors and erasures, each error counts as two erasures. For example for our case, (255, 223) RS code over GF(2^8), the code can handle up to 16 errors or 32 erasures or various combinations such as 8 errors and 16 erasures.

2.2 Sender and Receiver

The main entities of our technique are sender and receiver algorithms. These will reside as a part of new multicast protocol above UDP layer. The sender routine exists at the subnet router of sender of the video or voice... session or one of the intermediate routers (Designated Receiver-DR [32]). The receiver routine exists in the corresponding router of the subnet where the receiving host is located. The scheme we have used is receiver initiated so that instead of acknowledgments, negative acknowledgments (NAKs) are sent by receivers to the sender when either the number of lost original packets are more than received parity packets or there is a decoder failure due to the fact that number of erasures and errors exceed correcting capability of the RS code.

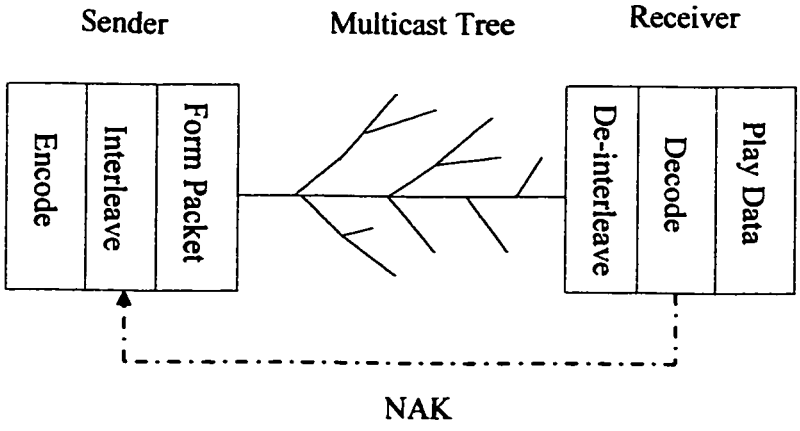


Figure 2.1: Sender and one Receiver in the Multicast Network

Figure 2.1 shows the sender and one receiver in the multicast network and their main building blocks.

2.3 Packet Formats and Preparation

Two main types of packets are transmitted (Figure 2.2). First, Data and parity packets transmitted by sender. Second, NAKs sent by receiver to request retransmission.

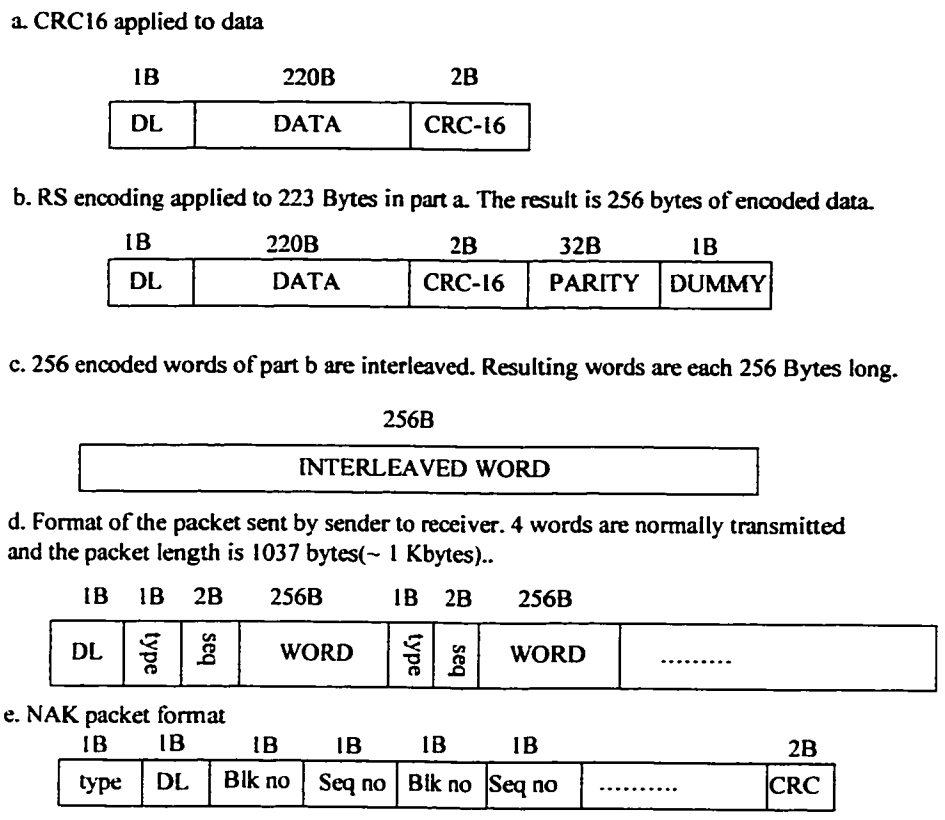


Figure 2.2: Packet formats for FEC/ARQ scheme

2.3.1 Data and Parity packets

Following steps are done in the sender to prepare and send the packets containing data and parity information to the receiver.

2.3.1.1 Encoding

As in Figure 2.2 data is split into groups of K-3 (220) bytes each. One byte is allocated for indicating the Data Length (DL. Number of bytes of data being transmitted in each group is always K-3 except for the last group of data that is less than or equal to K-3). A two Byte Cyclic Redundancy Check (CRC-16) [33] field is also formed at the end of data group to make a total of K (223) bytes. The resulting 223 bytes are fed into (255, 223) RS

encoder and the output is an N (255) bytes encoded word. Since the RS code used is systematic, the first K (223) bytes of the encoded word will be the same as input bytes (including Data, DL and CRC) and the last N-K (32) bytes will be redundant (parity) bytes. A one byte of dummy data is added to obtain a 256 byte encoded word.

2.3.1.2 Interleaving

Block interleaving is next applied to a record of 256 such encoded RS words, to yield a corresponding interleaved record [34]. The resulting first interleaved word of such record consists of the first byte of each of encoded words before interleaving. The 256^{th} interleaved word is similarly the concatenation of bytes number 256 of each pre-interleaving word. Now the first 223 interleaved words contain the data and the next 32 interleaved words include the parity information. We call each set of 256 interleaved words that corresponding to 256 encoded words a *block*.

It is also possible to have a smaller depth interleaving e.g. 128 in which case the first interleaved word will contain the first byte of 128 pre-interleaving words followed by the 127 interleaved words each consisting of one byte of 128 pre-interleaving words. Figure 2.3 shows an encoded block of data before and after block interleaving at sender.

2.3.1.3. Preparing packets

In order to prepare packets to be transmitted, a two bytes interleaved word sequence number field is added to each interleaved word, this is preceded by one byte of type field denoting an interleaved word. The sender routine groups up to 4 interleaved words in one UDP data unit. This makes a data unit length of approximately one Kbyte. Larger number of interleaved words such as eight can also be grouped to make the data unit.

At the beginning of the data unit (packet to be transmitted), a data length field is included. This will enable the receiver software to know how many words are included in the data unit while the constant length of 259 bytes, the type, and the sequence fields of each word will enable correct parsing and processing at the receiver side.

All the packets that are transmitted have the same configuration whether they are retransmitted packets (repair) or not. The difference is that the selected words can be either interleaved words or encoded words (in case of decoder failure in the receiver due to the fact that number of errors plus erasures has exceeded decoding capability of RS decoder). Much of the processes above are done offline in the sender, i.e. the whole

MPEG or voice file...etc. is encoded, interleaved as above and stored before starting the multicast in real time. The process of adding interleaved words and preparing data units is executed in real time once the multicast session starts.

0-0	0-1	0-2	....	....	....	....	....	0-254	0-255
1-0	1-1	1-2	....	.....	.....	.....	....	1-254	1-255
2-0	2-1	2-2	....	....	....	....	....	3-255	2-255
....	....	....	....	....	....	....	....	....	....
....	....	....	....	....	....	....	....	....	....
....	....	....	....	....	....	....	....	....	.....
....	....	....	....	....	....	....	....	....	....
....	....	....	....	....	....	....	....	....	....
254-0	254-1	254-2						254-254	254-255
255-0	255-1	255-2						255-254	255-255

a. A block of encoded words. Each row is an encoded word and each cell is a Byte.

0-0	1-0	....	....	....	....	....	....	254-0	255-0
0-1	1-1	....	....	....	....	....	....	254-1	255-1
0-2	1-2	....	....	....	....	....	....	254-2	255-2
....	....	....	....	....	....	....	....	....	
....	....	....	....	....	....	....	....	....	
....	....	....	....	....	....	....	....	....	
....	....	....	....	....	....	....	....	....	
....	....	....	....	....	....	....	....	....	
0-254	1-254	....	....	....	....	....	....	254-254	255-254
0-255	1-255	....	....	....	....	....	....	255-254	255-255

b. The same block in a (above) after block interleaving. Each row is a concatenation of same byte number of 256 pre-interleaving encoded words in a.

Figure 2.3: An encoded block of data a. before and b. after interleaving at the sender.

2.3.1.4 Burst packets

After transmission of each set of 256 interleaved words (window) a burst pulse containing for example 5 short packets of each two bytes length, is transmitted. First byte of each packet is the type and the second byte is the sequence number of the packet. By receiving any of the above packets, the receiver will realize that all 256 words belonging to the current window have been transmitted and in case it has been waiting to receive more words, it will stop it and will start de-interleaving and decoding received interleaved words.

2.3.2 Negative acknowledgement (NAK)

Upon receiving interleaved words and in case the number of received words is less than K (223), the receiver will request retransmissions (repairs) by sending a negative acknowledgement (NAK) packet to the sender.

The negative acknowledgement NAK format is shown in Figure 2.2. A one byte type field indicates whether it is a request for interleaved or encoded words. A one byte field denotes the data length of the NAK message, i.e. the number of requested words. The following fields give the sequence number of interleaved or encoded words as asked by the receiver. A final one byte CRC field is used for error checking at the sender.

2.4 Sending and Receiving Processes and Programs

All the codes are written in C programming language and run on Linux/Unix operating system. The length of the code for sender and receiver is 600 and 2400 lines respectively. In order to communicate between sender and receiver PCs, UDP socket programming [35] is used. At startup, receiver is ready to receive packets from sender. The sender sends packets to receiver IP address and UDP port. Upon receiving a packet, receiver extracts the words and stores them into relevant arrays depending on their type.

Using a Unix Signal: we use a type of signal called asynchronous I/O with non-blocking socket calls" in order to let the receiver perform other tasks when there are no packets on its UDP socket.

Following are some details about signals. Generally signals provide a mechanism for notifying programs that certain events have occurred, for example, the user typed the

"interrupt" character, or a timer expired. Unix has dozens of different signals with various triggering events. The signal we have used is SIGIO and its triggering event is "socket ready for I/O". Some of the events (and therefore the notification) may occur asynchronously, which means that the notification is delivered to the program regardless of where in the code it is executing. In our case, the operating system informs the program when a socket call will be successful. This way the program can spend its time doing other work (such as de-interleaving, decoding and preparing NAKs, in case of receiver) until notified that socket is ready for something to happen. This is called asynchronous I/O, and it works by having the SIGIO signal delivered to the process when some I/O–related event occurs on the socket.

In our program the receiver is able to perform other tasks when there are no packets form the sender on the socket. After creating and binding the socket, instead of calling recvfrom() and blocking until a datagram arrives, the SIGIO signal is delivered to the process, triggering execution of the handler function. The handler function calls recvfrom(), extracts words from the datagram (packet), stores the words into appropriate arrays for further use by the program, and then returns, whereupon the main program continues whatever it was doing.

Complete details about signals and UDP sockets are found in TCP/IP socket programming texts [36].

2.4.1 Receiving Process

When the receiver program is initiated, receiving word arrays are checked for new interleaved words (As mentioned above, new words are stored in those arrays after a call to SIGIOhandler and receiving packets on UDP socket). If there is a new interleaved word its sequence number is compared with the expected one so that missing words are detected. If word sequence number is greater than what is expected, there is a loss. Then the word is de-interleaved into one byte of each 256 encoded word. The process is repeated until there is no more new received word. In this case the receiving process for the current block of words (256) is finished provided that one of the three followings occurs. Otherwise program checks arrays for new words and the loop repeats itself.

1. Last interleaved word of the current block is received (seq. no. 255)

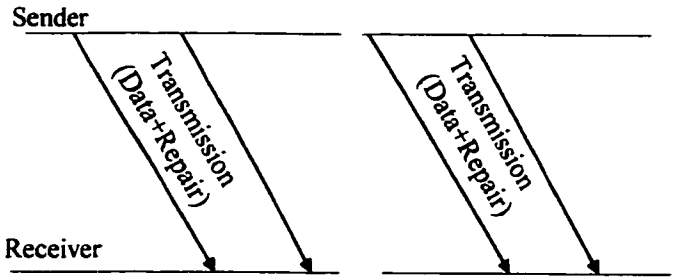
2. A word from the next block has arrived.
3. End of block burst packet is received.
4. Timer for receiving current block words expires.

Upon finishing the receiving process number of received bytes of encoded words (q) are checked. If q is greater than or equal to K (223) all data bytes are recovered. According to Figure 2.4.a, if no data byte is lost no decoding is required and data is already available.

But if some data bytes are lost the decoder will be called and will recover the lost bytes if there is no error in the received bytes (Figure 2.5). In case 2*(number of errors) + (number of erasures) is greater than N-K the decoder will fail and consequently relevant encoded words will be requested by sending NAK.

In case q is less than K, a NAK containing sequence number of (K-q) packets are sent to the sender. The state of the receiver changes to first request and the timer for new state is started (Figure 2.5). After sending NAK the program checks if there are retransmitted interleaved words in the relevant array (retransmissions arrived on the socket). Arrived words are de-interleaved, number of available bytes of each de-interleaved word is compared with K and decision for decoding or sending another NAK is made.

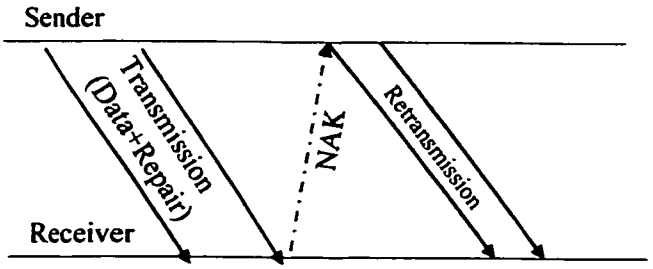
Flowchart of the main receiver program used in our experiments in ETS laboratory, Montreal gives more details about receiver. According to the code, the receiver maybe in one of the two states. The first state is for receiving and processing fresh packets (i.e. packets that are transmitted by the sender for the first time) and it is called "block fresh receive state". See Figure 2.6. If the current block of packets can be processed and completed without a need to retransmissions the receiver remains in this state and starts receiving and processing next block of packets, otherwise the receiver state changes to " block request state" and it starts processing retransmitted packets. The flowchart for this state is shown in Figure 2.7 as part 2 of flowchart.



a. No lost data. Decoding not required. Data ready as Received.

b. No of Lost Packets less than / equal to N-K. Decoding recovers lost data.

Figure 2.4: Receiving Process. The two cases that data is recovered without sending NAK and receiving retransmissions.



No of lost Packets greater than N-K. NAKs are sent.
Requested Packets are retransmitted.Decoder recovers lost data.

Figure 2.5: Receiving Process. The case when NAKs and retransmissions are required to recover data.

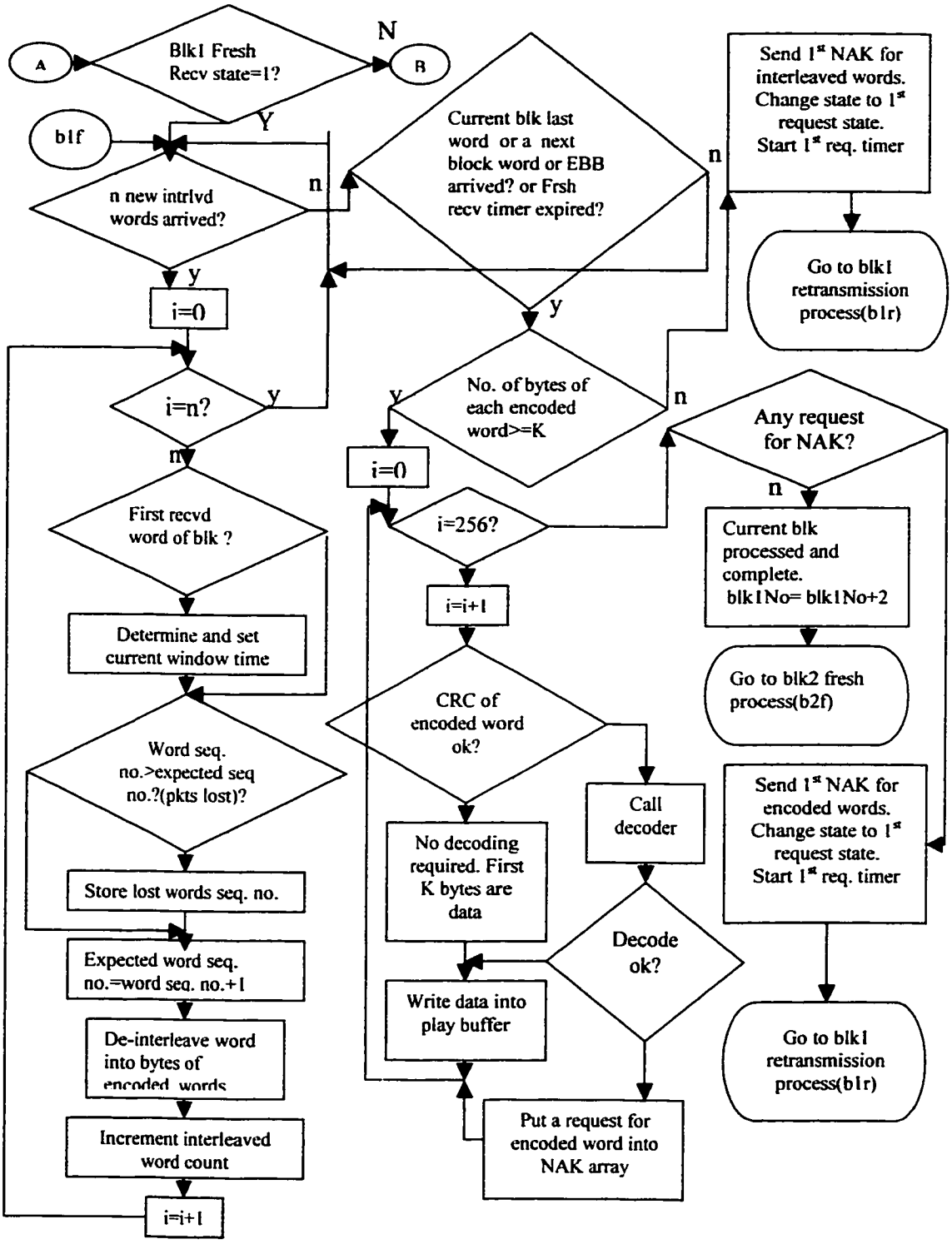


Figure 2.6: Part 1 of Receiver flowchart used in experiment. Block1 fresh receive process (b1f). Flowchart for b2f is similar.

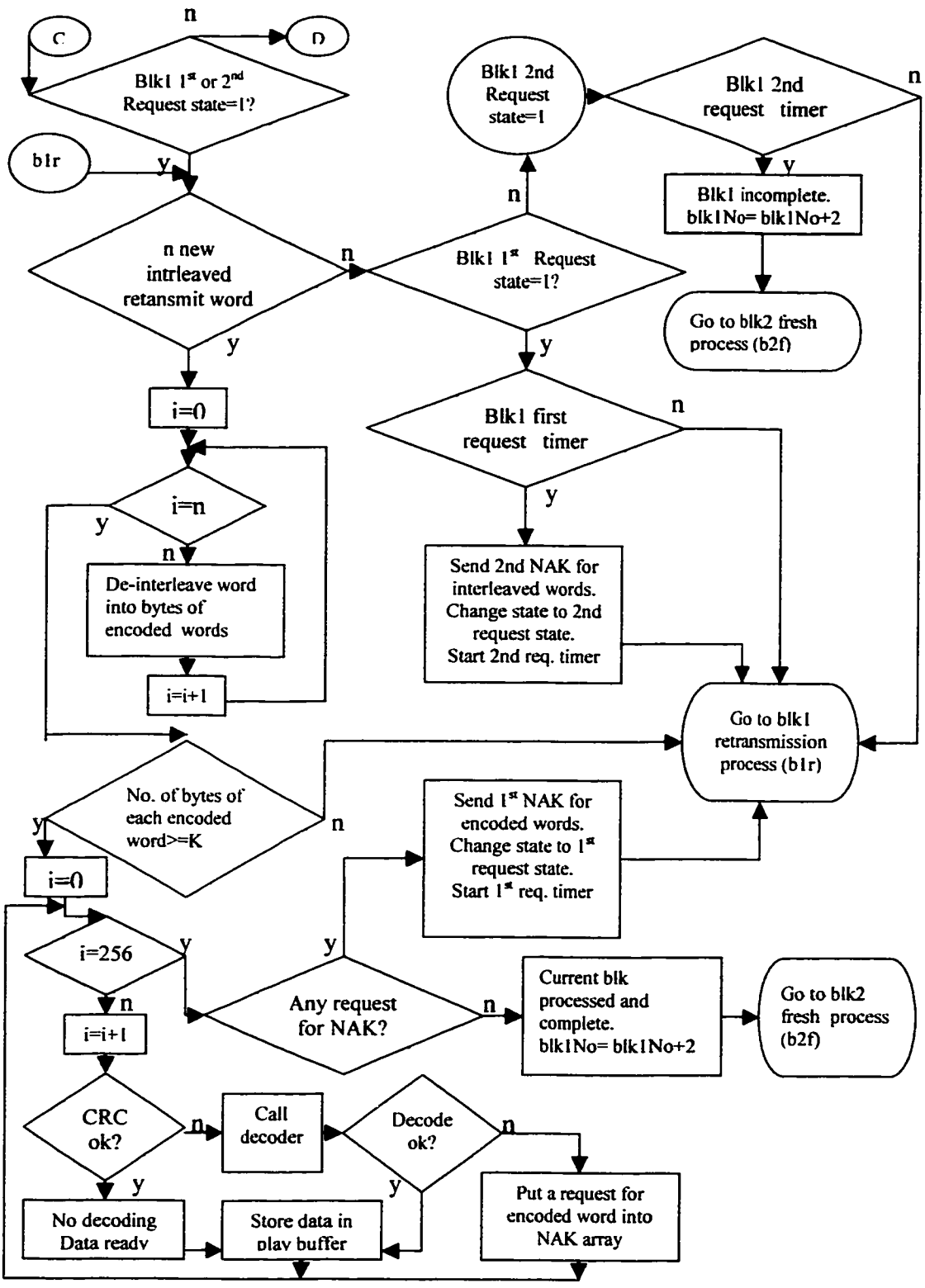


Figure 2.7: Part 2 of Receiver flowchart used in experiment. Block1 retransmission process (b1r). Flowchart of b2r is similar.

2.4.2 Window Mechanism

We define time T to be the total time to receive and process a block of words (as stated before, each block of words contain 256 words and each word is 256 bytes long), i.e.

$$T=RT+PT \tag{2.1}$$

where RT stands for Receive Time of one block and PT is its max Process Time. PT includes all actions taken in the receiver to recover erroneous or lost packets like de-interleaving, CRC checking, RS decoding, sending NAKs and receiving retransmissions. In Figure 2.9 receive and process details are illustrated. Among those actions, RS decoding is the most time consuming one specially if number of erasures is N-K (i.e. max number of erasures that can be recovered). In our experiment with two 600 Mhz PCs and using two Ethernet (10Base–T) cards max decoding time was about 90% of the total Process Time (PT). So the max process time of a block and the flow rate (reciprocal of PT) was determined. Since flow rate is also equal to reciprocal of RT,

$$PT=RT \tag{2.2}$$

Also because the sender is assumed to send packets continuously and at fixed rate, it means that next block of words generated by sender are being received while the receiver is processing the current block. Words of Next block are stored and will be processed after current PT is finished. Figure 2.8 shows receive, process and play time of various blocks. For example, for block number n, receiver window w (n) is open. During this time window, block n is received and processed and block n+1 is received and stored but words belonging to other blocks are discarded (if received). After block n processing is finished the window slides forward so that block n+1 can be received and processed and also block n+2 is received.

2.4.3 Sending Process

The same signal as in the receiver is used in the sender, and in case of receiving a datagram (NAK) the SIGIO signal is delivered to the process, triggering execution of the handler function. The handler function calls recvfrom(), sequence number of requested words are extracted from NAK packet and put in the relevant queue, and then after it returns, the main program continues to do what it was doing before.

As mentioned before, encoding data and interleaving the encoded words are done offline i.e. they are prepared and stored in files prior to starting the session. So the sender program opens those files and uses their contents to build the packets to be transmitted. At start of program, when there is no request (NAK) from the receiver, sender continuously prepares packets i.e., one packet from 4 consecutive interleaved words and transmits them at a fixed predetermined rate to receiver. After transmitting every 64 packets (256 words) a few End of Block Bursts (EBB) are sent to inform the receiver of the block end. Sending the packets are continued until a NAK is received asking for retransmission of some words. Sender stops sending normal packets (hereafter called fresh packets). Block number and Sequence number of requested words are stored in a queue and requests are served based on FIFO (first in, first out). Sender transmits requested words as fast as it can so that retransmissions are received fast and data recovery is done as quickly as possible. When the queue is empty the sender continues to transmit the rest of fresh packets until it receives another NAK...

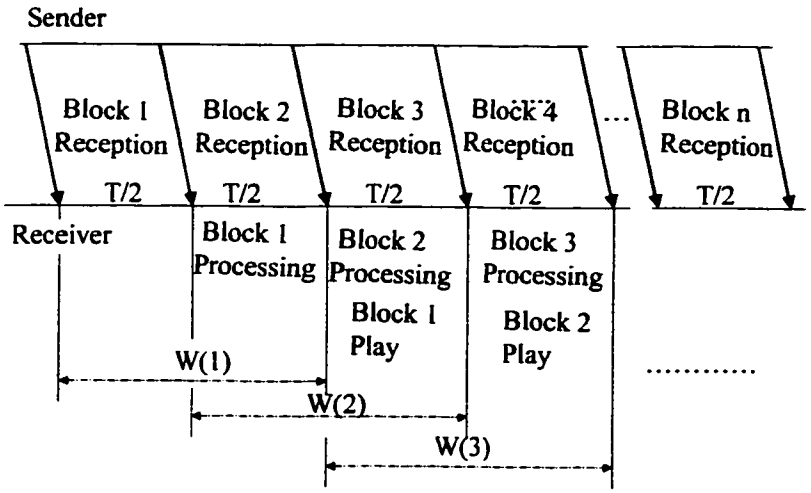


Figure 2.8: Sliding window mechanism at receiver

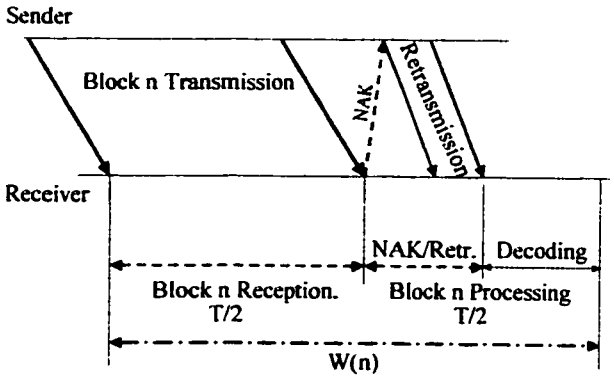


Figure 2.9: Reception and processing details at receiver

2.5 Experimental Details

Experiments were performed in ETS (École de Technologie Supérieure) network laboratory in Montreal.

***Experiment setup*:**

Two 600 MHz PCs were used, one as receiver and the other one as sender. Both PCs were equipped with Ethernet 100BASE-T (100Mbps capacity) interface cards for communication between each other. In order to test and evaluate the performance of FEC /ARQ codes on the two PCs, network impairments had to be applied to the packets that are transmitted and received. A third PC (733 MHz workstation) equipped with network impairment emulation software called IP WAVE was used. Two 10BASE-T network interface cards were already installed on the PC for communication. Each interface port (can be called a gateway) of the IP WAVE PC can be attached to a separate sub-network (in our case sender or receiver). The IP packets are forwarded from each sub-network to another, with the IP WAVE adding its impairments in the forwarding process.

Six different impairments that can occur in networks may be applied to IP packets using this instrument. Various kinds of impairments can be applied independently. Packets may be duplicated, delayed and fragmented or out of order errors can be applied. Errors can be injected into selected packets. Packets may be dropped in a periodic, random and bursty manner. Packets could be impaired at speeds up to 10 Mbps for each

direction of flow. The system setup is shown in Figure 2.10. The impairment we have applied is dropping packets and in one case we have injected errors into a certain percentage of packets.

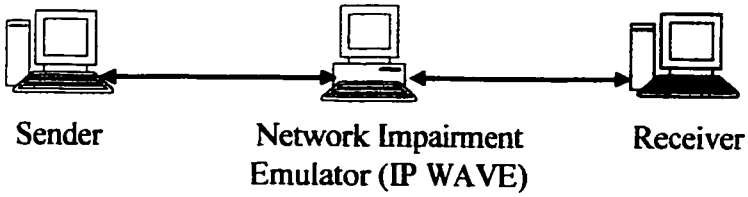


Figure 2.10: Experiment setup at ETS network laboratory

2.5.1 Experimental Procedure

Various performance parameters of RS/ARQ scheme were measured. For each test, transmission rate and random packet loss were specified and applied. Transmission rate was selected by changing the time interval between successive packet transmissions by the sender. The transmission rate was changed from 1000 Kbps to 2400 Kbps, at 100 Kbps steps. As mentioned in the introduction to this chapter, above flow rates are for packets containing data, parity and other redundant fields. Actual flow rates for data are less. For instance, for a value of 2Mbps the actual flow rate for data is 2*(220/256)=1.7 Mbps. Random packet loss was selected by changing the settings on impairment emulator. Packet loss was changed from 0 to 10 %, at 1% steps. For each transmission rate one graph of the desired parameter versus flow rate could be drawn. By changing transmission rate as the second variable, three-dimensional graphs of the parameter versus transmission rate and random packet loss are plotted.

2.5.2 Measured parameters and explanation of results

2.5.2.1. Delay

Delay for each word is the time it takes for a word of data (220 bytes) to be processed at sender (encoding, cyclic redundancy checking, interleaving, preparing packet, delivering it from application to UDP port), transmitted over the link to the receiver and processed in the receiver (de-interleaving, CRC, decoding) until decoded data is ready to be played. Word delay is calculated from equation below:

$$\Delta_i = t_{i+1} - t_i \tag{2.3}$$

t_i is the time when encoding the word starts and t_{i+1} is when the word is ready to play after processing at receiver. The average delay $\bar{\Delta}$ (mean) of the transmitted words is calculated from

$$\bar{\Delta} = \frac{1}{N}\left(\sum_{i=1}^{N} \Delta_i\right) \tag{2.4}$$

N is the number of words (no of tests made at each specific transmission rate and packet loss).

Delay variance (σ_Δ^2) is a measure of spread around the average and is given by

$$\sigma_\Delta^2 = \frac{\sum_{i=1}^{N}\left(\Delta_i - \bar{\Delta}\right)^2}{N-1} \tag{2.5}$$

Figure 2.11, Figure 2.12 and Figure 2.13 give average delay ($\bar{\Delta}$), delay variance and standard deviation versus flow rate and packet loss respectively. As seen in the graph for a given flow rate, average delay increases with packet loss increase. When there is no packet loss average delay is minimum (i.e. 1ms). This small delay is mostly due to encoding, transmission and CRC. Decoding is not done in this case since there is no packet loss. Delay variance and hence standard deviation is zero indicating that delay of all words are the same. With packet loss increase both average delay and variance increase. As random loss increases more words need to be decoded and less words are recovered only with CRC being done. The more the number of lost bytes in a word (i.e. interleaved words in a block) the more time it takes for RS decoder to decode and recover the lost bytes. Decoding delay is the largest delay component and as stated previously, it

increases linearly with number of lost bytes. For the RS code used in our experiment, max decoding time (Δ_{max}) for N-K erasures is two times the minimum decoding time (Δ_{min}, when no of erasures is 0). Taking above into consideration, decoding delay (Δ_{dec}) for q number of erasures can be calculated from the following:

$$\Delta_{dec} = q\frac{\Delta_{min} - \Delta_{max}}{N-K} + \frac{\Delta_{max}N - \Delta_{min}K}{N-K} \tag{2.6}$$

For $\Delta_{max} = 2 * \Delta_{min}$ measured in experiment for our decoder, we have

$$\Delta_{dec} = \Delta_{max}\frac{(N-0.5K)-0.5q}{N-K} \tag{2.7}$$

At higher losses (around 10%), in some cases lost information is more than correcting capability of the encoder so NAKs are sent and retransmissions are received causing additional delay. By increasing the packet loss it is seen that ratio of standard deviation to average delay on a given flow rate slightly increases, meaning that at lower losses word delays are less close to the average.

For a given packet loss, by decreasing the flow rate, average delay increases. The obvious reason for that is slower transmission.

Here we give the reasoning for large values of delay variance in Figure 2.12. We consider 2000 Kbps transmission rate and 10% loss for which average delay and delay variance in Figures 2.11 and 2.12 are about 25 ms and 8000 respectively. In this case, the delay for packets that are received (90% of whole packets of a block) is only 1 ms that is mostly due to encoding, transmission and CRC. These are ready to play as soon as they arrive at receiver. The delay for lost packets (10%) consists of two components (Figure 2.9): 1) the difference between generation time of each lost packet and the time when block reception period is finished; 2) block processing time that includes decoding time and may also include NAK sending and retransmission times if losses are more than N-K. For above transmission rate total window time T will be about 500 ms and the delay for lost packets will be between 250 and 500 ms. Hence, with average delay of about 25 ms and wide spread of packet delays around the mean value, from 1 ms (for 90% of the packets) up to 250-500 ms (for 10%) above mentioned value for delay variance is reasonable.

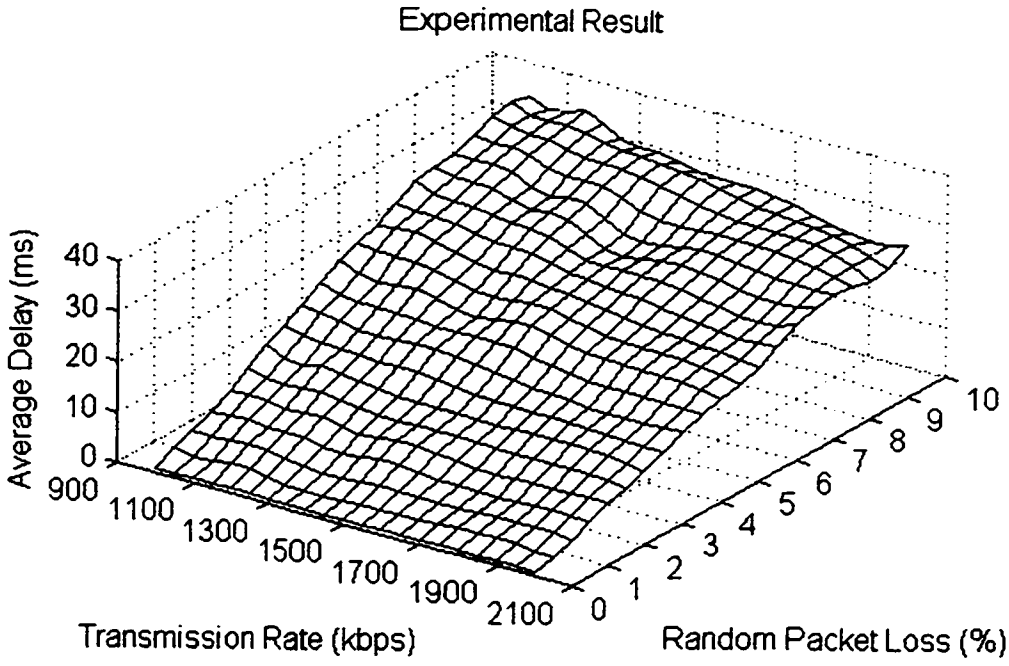


Figure 2.11: Average Delay versus Transmission rate and Random Packet Loss

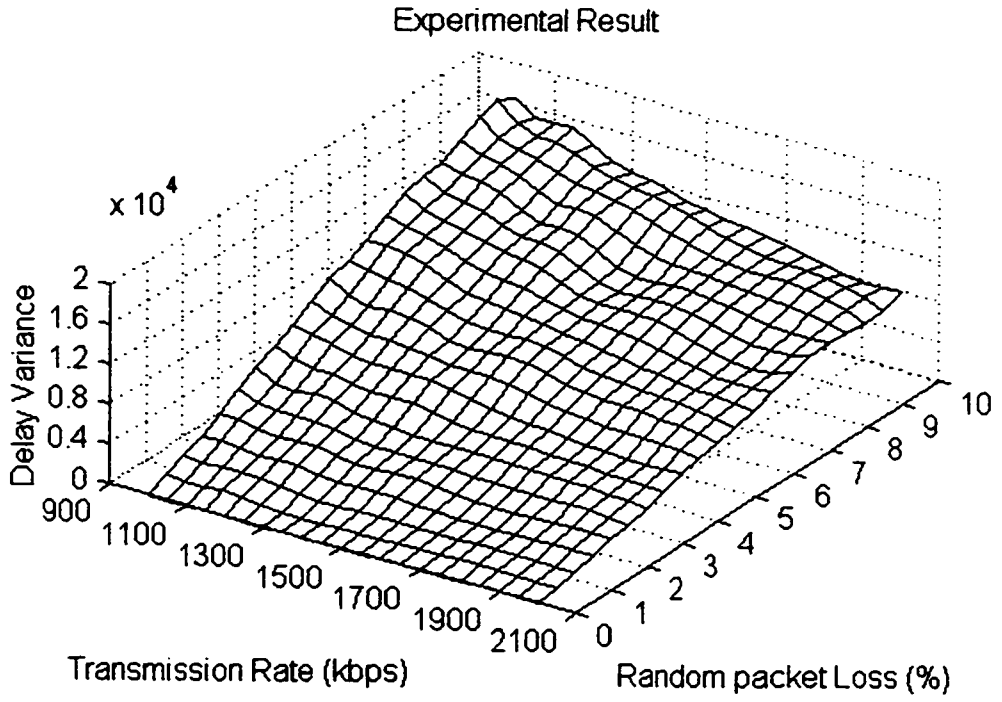


Figure 2.12: Delay Variance versus Transmission Rate and Random Packet Loss

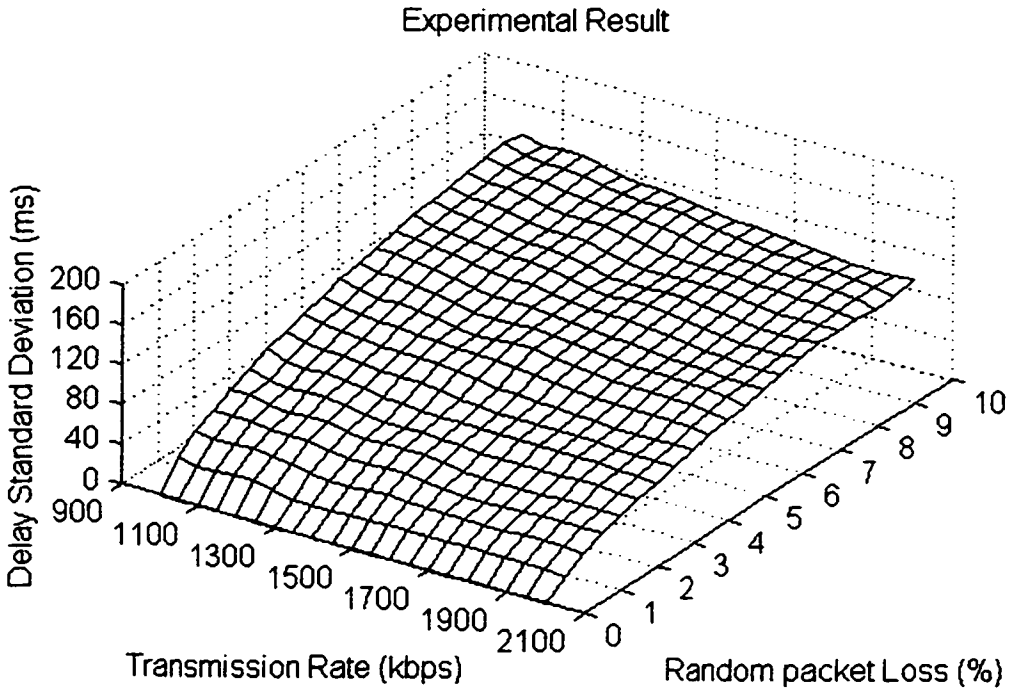


Figure 2.13: Delay Standard Deviation versus Transmission Rate and Random Packet Loss

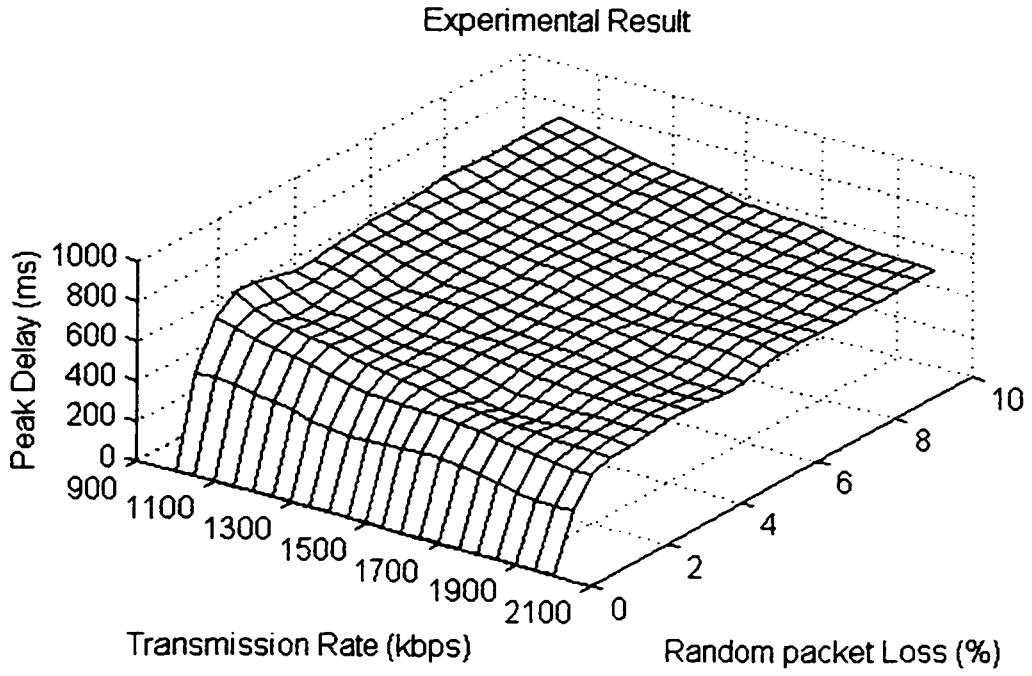


Figure 2.14: Peak Delay versus Transmission Rate and Random Packet Loss

Peak delay of the words versus transmission rate is shown in Figure 2.14. At 2000 kbps that is the maximum satisfactory flow rate of our design, the maximum peak delay is 485 ms. Referring to Figure 2.8 the maximum delay for a word in a window is T that is inverse of flow rate. This delay is 500 ms for 2Mbps, then peak delay of 485 ms is less than the width of a window and we are on the safe side. Peak delays in excess of window width (i.e. 500 ms) will result in loss of next coming block's data. This peak delay (480 ms) consists of two parts (Figure 2.9), a block transmission time (250 ms) and a block processing time (235 ms). The components of processing time are (de-interleaving, CRC,

NAK, retransmissions and decoding). For our setup, max decoding time for one block (256 RS words) is 230 ms. For a given loss rate, peak delay increases with decreasing transmission rate as stated above for the other delay curves.

2.5.2.2 CRC and Decode

Figure 2.15 shows the number of times "only CRC is called" versus Transmission Rate and Random Packet Loss. In order to clarify the term "only CRC is called" following explanation is given.

When there is no missing packet in the received ones of a block of words, CRC error detection is done. If there is no error and since RS coding used is systematic decoding is not required and the first 220 bytes of the word is data and is played as it is. This is called the case when only CRC is called. Graph of percentage of those cases (when only CRC is called) in all received words are drawn in Figure 2.15. At zero packet loss always "only CRC is called" and no decoding is required. As random packet loss increases no of cases with missing packets also increases causing decrease of times "only CRC called". The amount approaches zero approximately at 10% packet loss. A complementary graph for above is given in Figure 2.16 where percentage of "no of times the RS decoder is called" is displayed versus transmission rate and packet loss. At 0% loss decoder is not called at all and "times the decoder is called" increases with packet loss until it is called almost all the times at 10% packet loss.

2.5.2.3 Data recovered by decoding

Data bytes may be received or recovered in three ways. First, some of them are received directly at first transmission of information. Second, if enough number of bytes are not received for decoding and NAK is sent, retransmissions may include data bytes (These are data bytes received by automatic repeat request (ARQ). Third, data bytes may be recovered by decoding. Figure 2.17 shows the percentage of data recovered by decoding out of total data recovered (both via second way stated above i.e. ARQ and third way that is by decoding). As it is seen in the graph data recovery by decoding is more than 80% at all measured points (except at zero packet loss where no decoding is required). It means that only less than 20% are recovered by ARQ. This is a measure of how efficient and advantageous RS decoding (FEC) is as compared to ARQ.

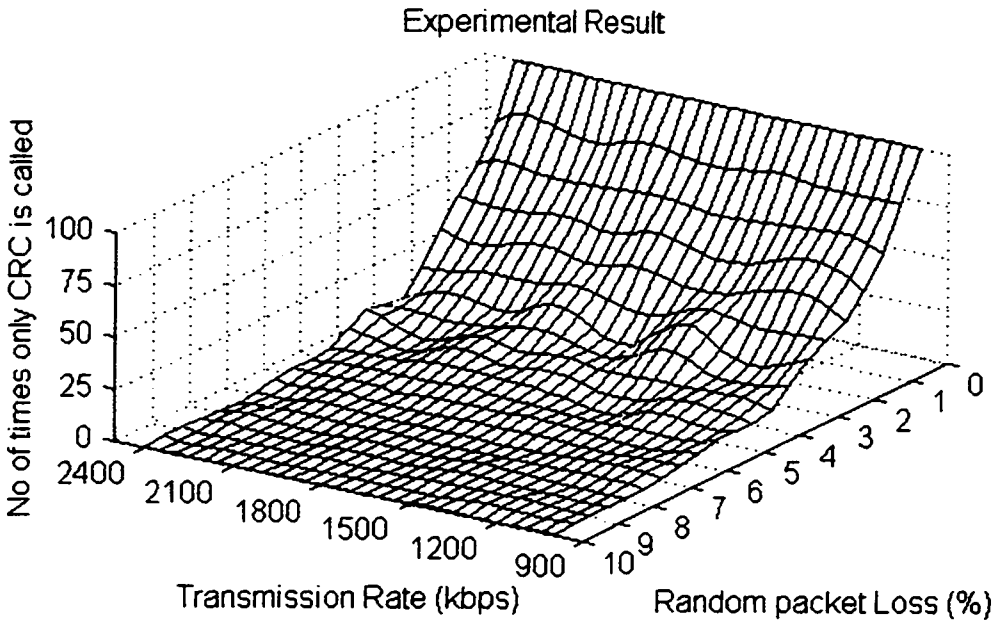


Figure 2.15: No of times only CRC is called versus Transmission Rate and Packet Loss

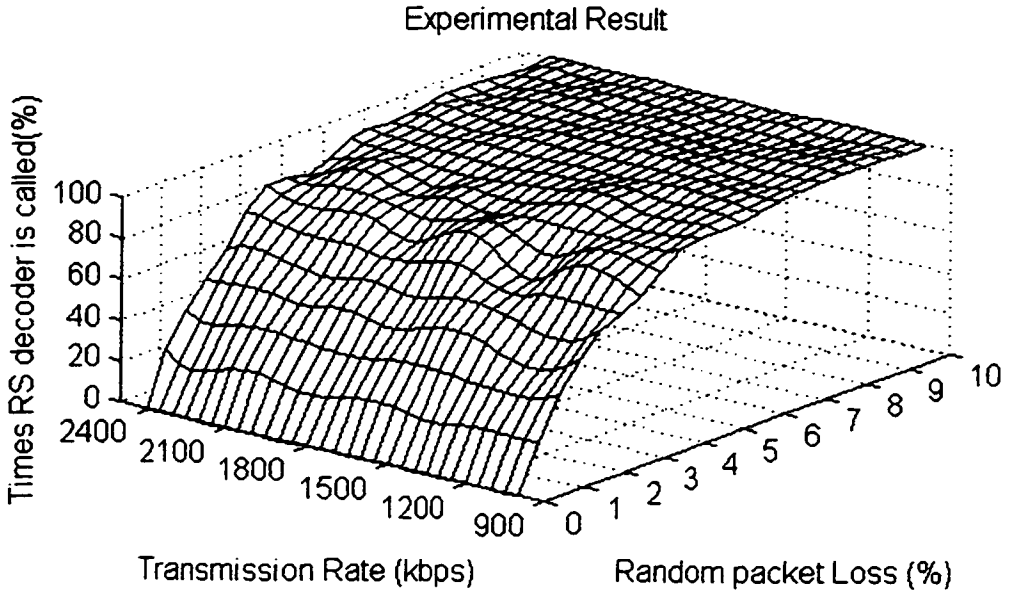


Figure 2.16: Times RS decoder is called versus Transmission Rate and Packet Loss

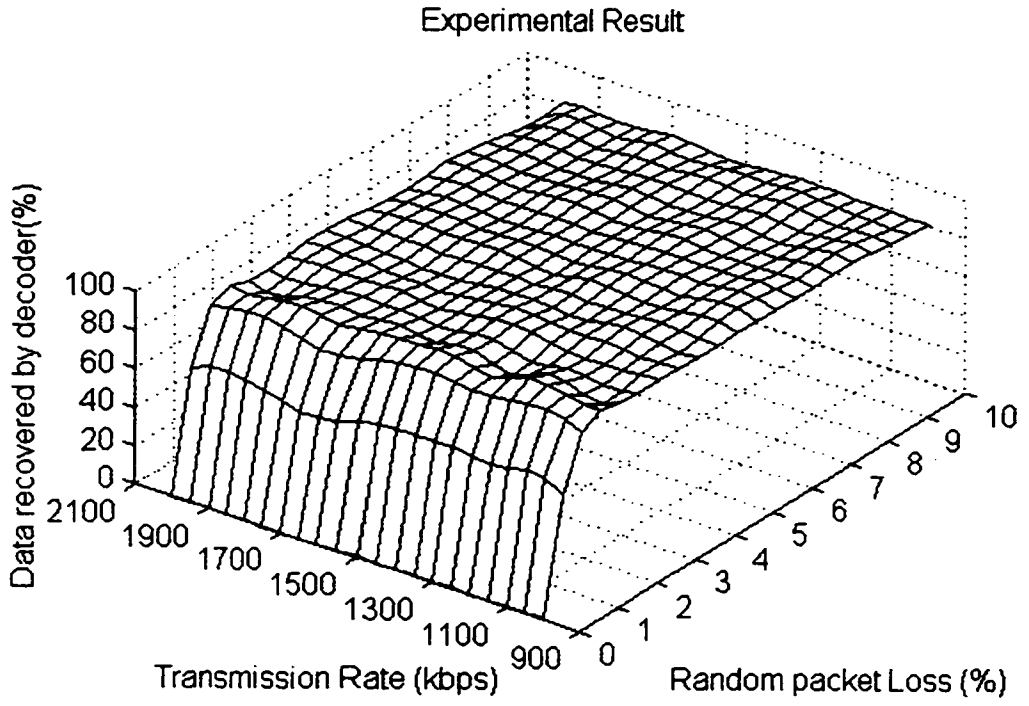


Figure 2.17: Data recovered by decoder (not via ARQ) versus Transmission Rate and Packet Loss

2.5.2.4 NAK and Retransmissions

In error recovery schemes based on ARQ, NAK is sent as soon as a single packet is found to be missing but in our FEC/ARQ scheme only one NAK is sent for a whole block (window) and it contains the sequence number of all bytes (i.e. interleaved/encoded words) that are required for recovery of lost information. Figure 2.18 shows the percentage of "windows generated NAKs" in whole blocks of the session. Number of NAKs sent for each block may be one or two. Second NAK is sent if all information requested through first NAK is not received and relevant timer expires. First and second NAKs are both counted and the total is displayed in Figure 2.18. As shown in the figure, number of NAKs increases with packet loss increase and the rate of change is positive. The reason is that for low packet losses a block rarely loses more packets than erasure correcting capability of the decoder (about 11%), while at higher packet losses that much loss is more probable.

As stated before, complete and satisfactory functioning of our implemented FEC/ARQ scheme is up to 2000 kbps flow rate. Above that rate up to a few hundred kbps more, it still functions well but some degradations are observed. More details will be given later. Looking at Figure 2.18 it is observed that number of NAKs increase when transmission rate exceeds 2000kbps and this increase is larger at high packet losses. For example at 10% loss, number of NAKs sent at 2300 kbps is 14% more than the maximum NAKs sent at rates below 2000 kbps. The reasoning is as follows. As mentioned in section 2.2.5.1, max measured time needed for decoding is 230 ms and total of other processing times is 5 ms, making a total of 235 ms, adding 25 ms as reserve, total processing time of a block (T/2) and as a result, minimum transmission time of a block is 260 ms. Since each block contains 256 words each containing 256 bytes, total number of bits to be transmitted in 260 ms is 256*256*8=524288. This is equivalent to a transmission rate of 524288/(260 ms) that is 2000kbps. At transmission rates above this rate and especially at high losses, processing at the receiver specially decoding won't be fast enough and processing time will exceed the allowed window time determined by transmission speed. As a result transmitted packets of next block will be lost for the duration of the time that decoding exceeds the window time. So in addition to packet loss caused by the network some more packets are lost because of fast transmission (slow

decoding) of packets. Consequently there will be more cases with losses exceeding erasure-correcting capability of the decoder and more NAKs have to be transmitted. Figure 2.19 is for NAKs sent for first time for a block and Figure 2.20 shows number of second NAKs sent. In both of the figures trends of increase in number of NAKs are the same as in Figure 2.18 (for total number of NAKs) and the same reasoning is valid.

Number of retransmitted words is also measured. The percentage is calculated according to following and the result is plotted and shown in Figure 2.21.

$$NR\% = \frac{RW}{TW} * 100 \tag{2.8}$$

NR is the number of retransmissions as a percentage, RW is the number of retransmitted words in response to NAKs and TW is the total number of transmitted words (excluding retransmitted words). NR in this figure follows similar variations as NAK graphs, i.e. at high packet losses and transmission rates number of retransmissions increase compared to lower losses and transmission rates, respectively.

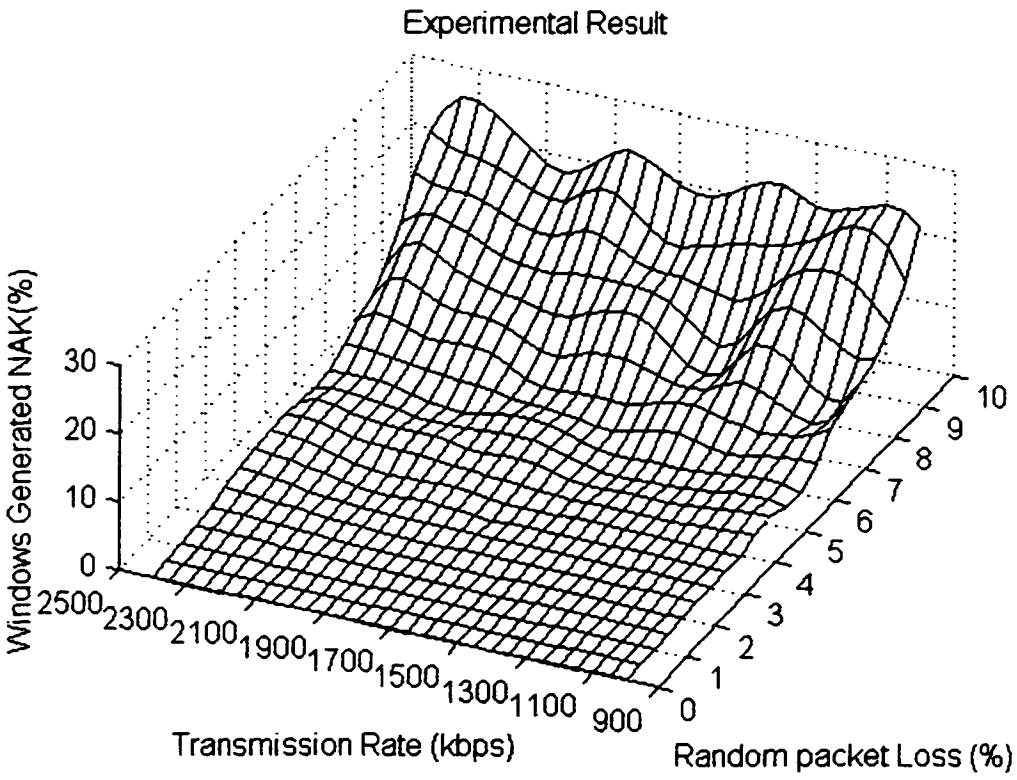


Figure 2.18: Windows Generated NAK versus Transmission Rate and Packet Loss

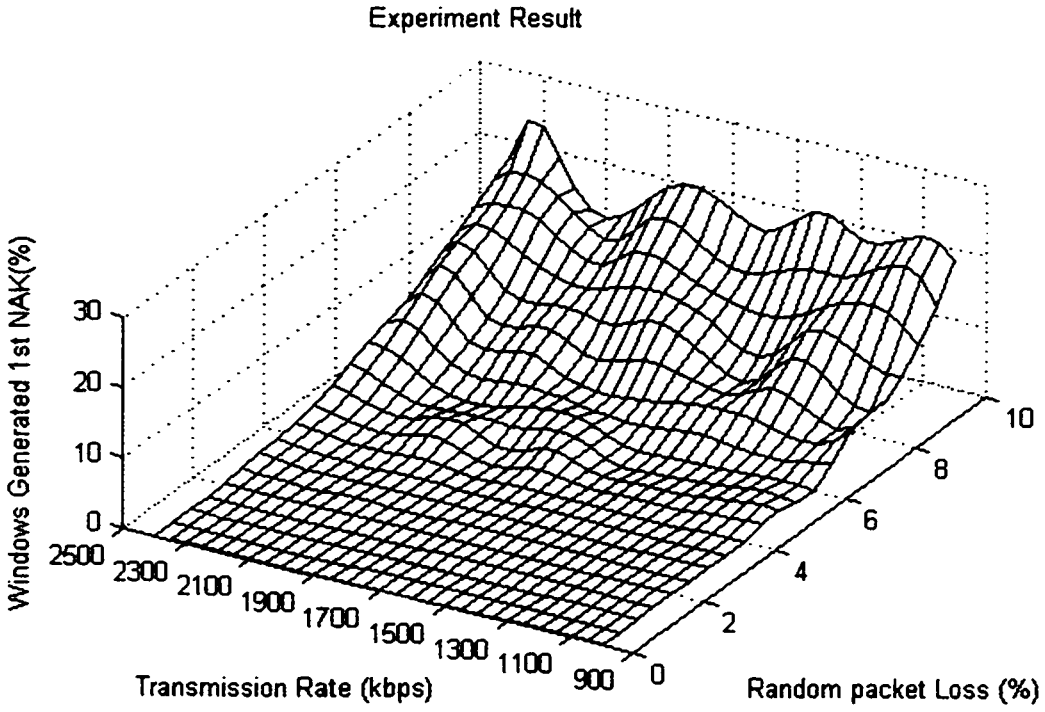


Figure 2.19: Windows Generated 1st NAK versus Transmission Rate and Packet Loss

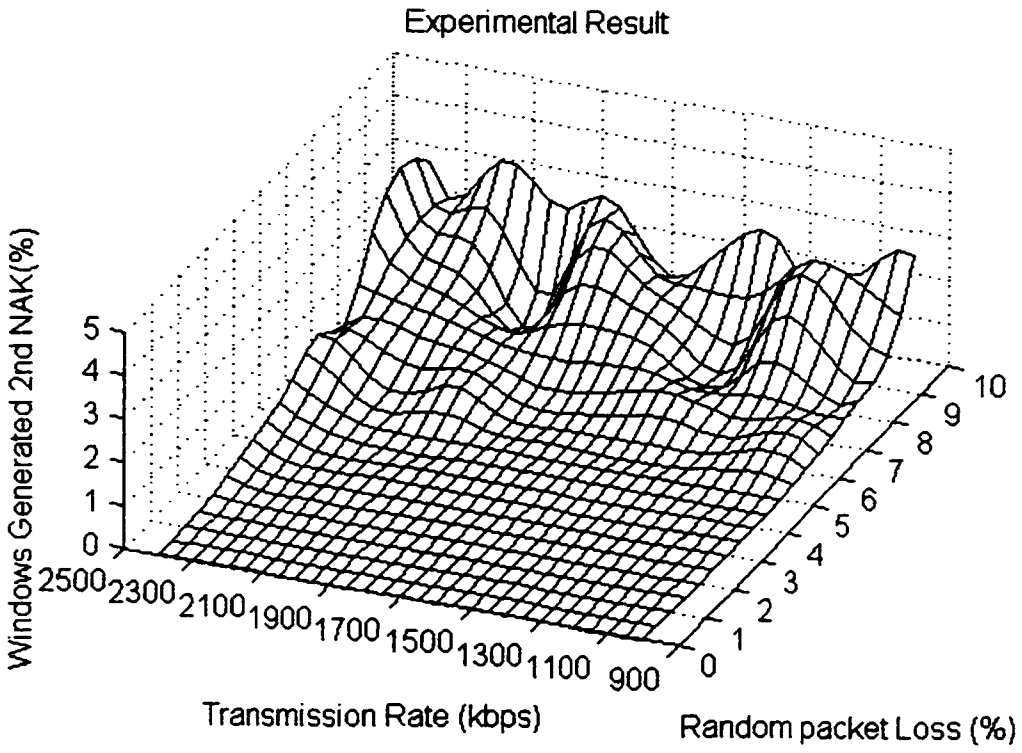


Figure 2.20: Windows Generated 2nd NAK versus Transmission Rate and Packet Loss

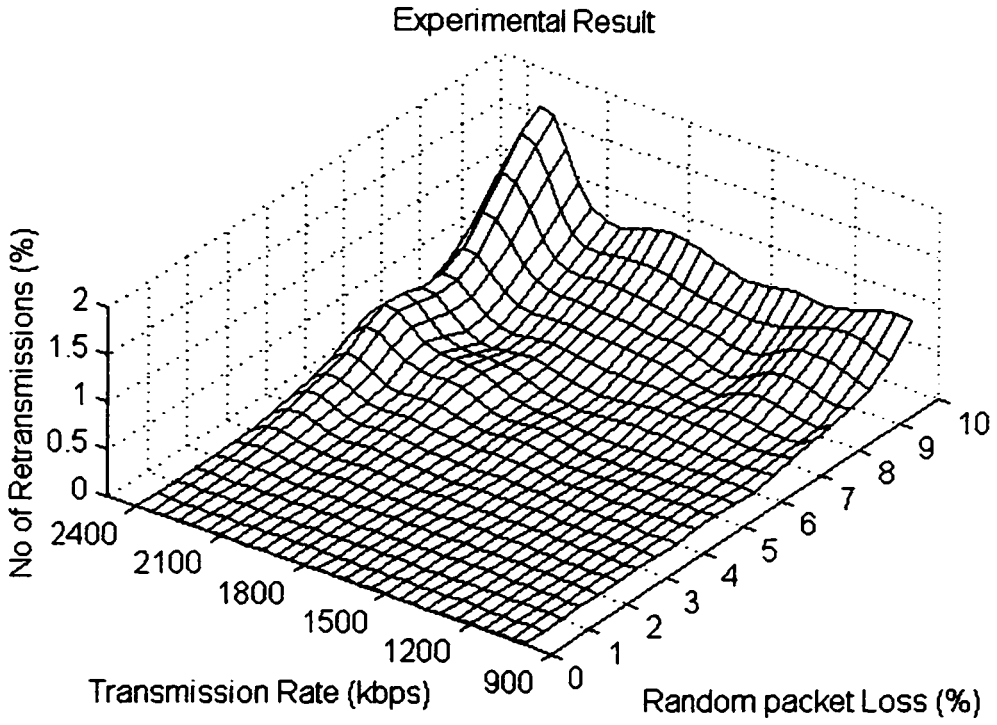


Figure 2.21: No of Retransmissions versus Transmission Rate and Packet Loss

2.5.2.5 Residual losses

Residual loss (RL) as defined below is plotted against transmission rate and packet loss in Figure 2.22.

$$RL(\%) = \frac{BNR}{BT} * 100 \tag{2.9}$$

BNR is the number of bytes that are not recovered (after RS decoding, ARQ, etc.) and BT is the number of transmitted bytes. As is shown in the graph, up to 1900 kbps, almost all losses are recovered and this scheme is completely reliable. Above 1900, some bytes remain un-recovered. Maximum value is 1.5% at 2400 kbps and 10% loss. The reason for above residual loss as stated for NAK section above is fast transmission of the data (slow decoding). This will cause sending more NAKs requesting for larger no of lost words. As a result there will be more cases that either some second retransmission packets are lost in the network or not arrived at receiver before second NAK timer expires. Consequently for some blocks or words enough repair information for decoding is not available and some lost bytes are not recovered. This is more pronounced at higher transmission speeds and losses.

2.5.2.6 Both Errors and Erasures

In the last test, in addition to dropping packets, errors were injected to some of them, measurements were made and behavior of the system was studied. Figure 2.23 shows number of retransmitted words versus transmission rate and packet loss while there are 1% byte errors in RS words. The same definition and calculation as for retransmitted words without errors is used. Comparing the two cases with (Figure 2.23) and without (Figure 2.21) 1% error it is seen that almost every-where (except at low packet losses where number of retransmissions are very low for both graphs) the case with error has considerably more number of retransmissions than the case without error. The reasoning for this is as follows.

As stated previously, 1% error for the decoder is equivalent to 2% additional erasure. As the first example for extra retransmissions, a case with 10% loss is considered. The decoder can marginally decode the word and recover the lost bytes but at the same loss condition and 1% error the decoder will fail because number of erasures plus two times number of errors i.e. 12% will exceed error and erasure correcting

capability of the decoder. As a result a NAK will be sent and the request will be re-transmitted.

As the second example, suppose a case where a NAK is sent (due to high number of lost packets). Upon receiving the requested repair information the decoder will try to decode (no of lost packets after receiving retransmissions is N-K) but in case of 1% error the decoder will fail. Now not only another NAK and retransmission (in response) is sent but also one more decoding is done. Transmission of extra NAK, retransmission and decoding will lengthen the process time that may cause some packets of next (block) to be dropped, and as a result more NAKs and retransmissions will be sent. This situation is much more pronounced when transmission time of a block is less than decoding time (i.e. high transmission rates such as 2000 kbps and more).

Third, even when the decoder succeeds in decoding, the time consumed for decoding is more than the case without error, thus increasing the possibility of insufficient processing time causing packet drops and more NAKs and retransmissions.

2.6 Conclusion

The results of the above tests show that our integrated FEC/ARQ scheme makes reliable communication between two PCs for flow rates up to 1.7Mbps (2Mbps without considering redundancy due to FEC) and losses up to 10%. This is achieved at extra cost of coding/decoding in the end systems. Coding delay in the sender can be reduced in different ways:

1. Pre-encoding the packets off-line and storing the encoded information on disk prior to transmission.
2. Using a more powerful machine at the sender.
3. Using dedicated hardware in the sender.

The receiver is simply equipped with its part of code and will start to play the extracted data after an initial start up delay of one window length (e.g. 500 ms for a flow rate of 2Mbps). It is also observed that error/erasure control feedback for this system is very small, since NAK is sent for each block of information (consisting of N words) instead of for each missing packet and only when packet loss rates are large (greater than N-K). Also number of retransmissions at worst case, i.e.10% loss and 2Mbps flow rate is

less than one percent of total transmitted information. This indicates that minor usage of extra network resources to achieve complete reliability. In case of burst loss, performance of FEC is improved by interleaving encoded words prior to transmission. Interleaving lets the sender spread the transmission of an encoded word over an interval that is longer than the burst loss length.

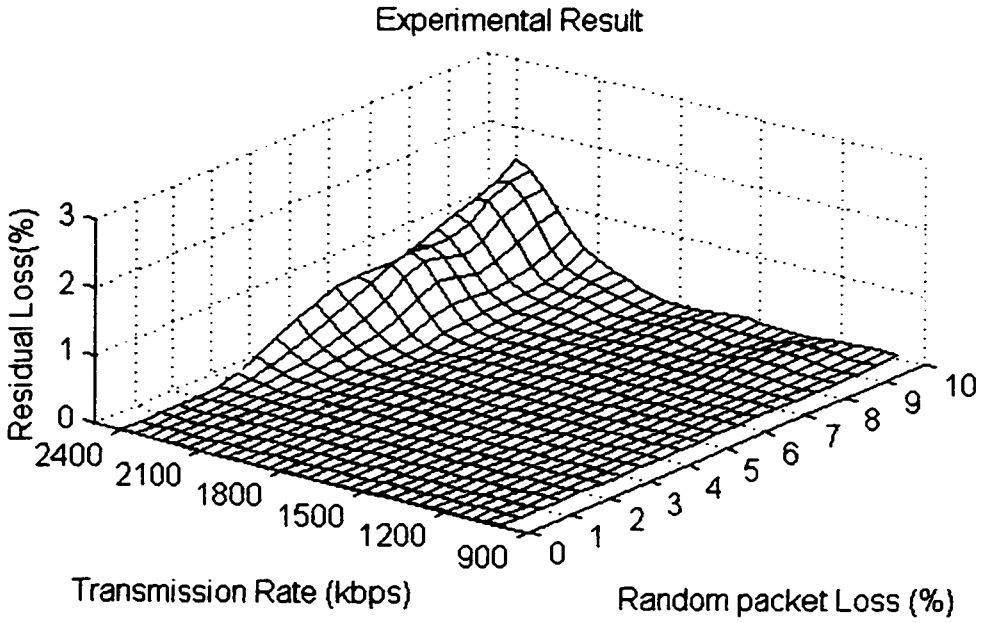


Figure 2.22: Residual Loss versus Transmission Rate and Packet Loss

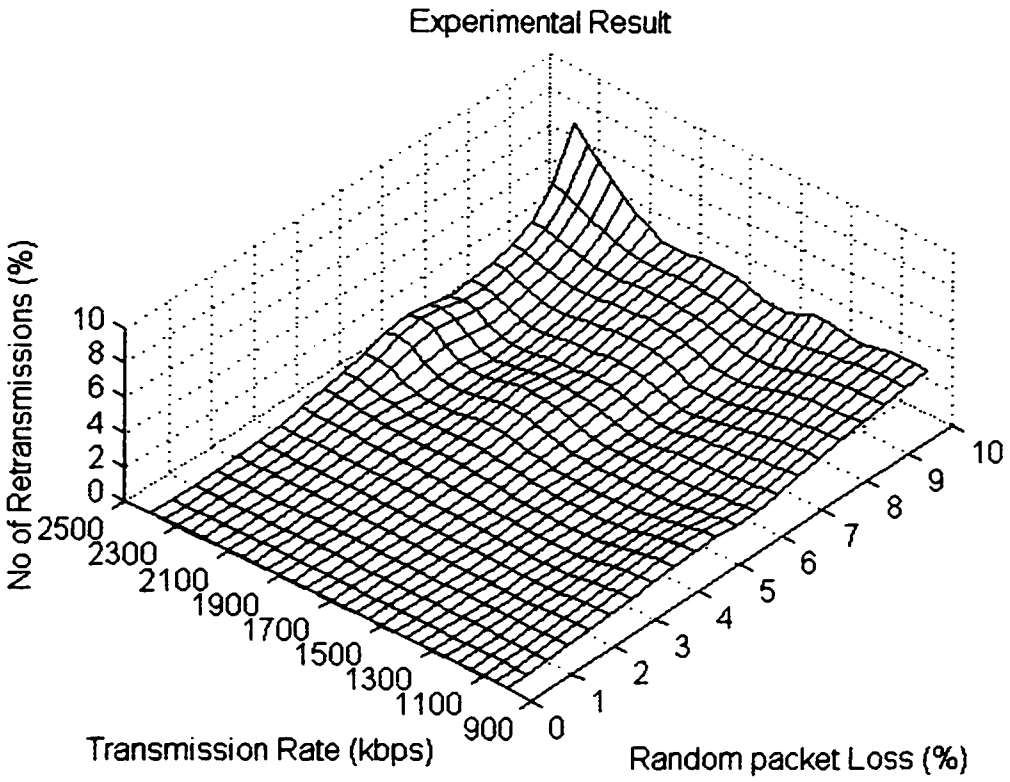


Figure 2.23: Retransmissions (while 1% byte errors in RS words) versus Transmission Rate and Packet Loss

Chapter 3

Simulation of QOS Multicast

We are going to simulate our QOS multicast scheme in this chapter. We consider a hypothetical network that covers 4 flows with different number of receivers, various topologies and different number of flows through routers. The code written for the simulation is in C programming language and run on windows operating system. It is over 10000 lines.

Sender, receiver and router algorithms for the proposed FEC/ARQ scheme are given in section 3.1. We also simulate a generic version of a reliable multicast protocol that uses only ARQ to recover lost packets. This protocol is explained in section 3.2. Section 3.3 discusses NAK avoidance mechanism used in the simulation.

In section 3.4 simulation results are given. Performance of FEC/ARQ scheme is studied and where necessary, the results are compared to ARQ scheme, with or without NAK avoidance. Four cases are considered. First, independent losses for homogeneous receivers are discussed followed by shared losses. Then simulation results of heterogeneous receivers are explained. The last case considers effect of queues in the performance of the FEC/ARQ scheme.

3.1 FEC/ARQ Scheme

In order to study the performance of integrated FEC/ARQ scheme in multicasting, a hypothetical network that covers four multicast flows is considered. It is assumed that multicast (packet distribution) trees are built and multicast groups are registered and established according to Figure 3.1 and will remain constant during simulation. The four multicast trees are chosen with different number of receivers, various topologies (shared, independent and mixed links) and different number of common routers between flows. Above is done so that the network covers most features of a real multicast network. Flow 1, 2, 3 and 4 cover 4, 6, 5 and 3 receivers respectively and also there are 13 routers in the network that receive packets of various flows and route them to next hops and eventually to relevant receivers. An iteration is a unit of time at which actions are taken at all elements of the network i.e. senders, routers and receivers and is equal to transmit time of a retransmission packet. In Figure 3.2 four multicast flows are separated from each other. Table 3.1 and Table 3.2 show multicast routes and routing tables for the hypothetical network.

3.1.1 Sender

As stated in chapter 3, n=(N+1)/4=64 encoded and interleaved packets, k=(K+1)/4=56 data and 8 repair packets, (~1 Kbyte each) per window are generated. At certain iterations, a fresh packet is generated depending on the ratio between transmission capacity of the link and sender flow rate. For example, if the link is 4 times faster (gR=4), fresh packets are generated once in every four iteration, i.e. iteration no. 1, 5, 9, 13... Sender will stop generating above-mentioned fresh (data and repair) packets as soon as a NAK is received. Generation of fresh packets is continued after requested packets are retransmitted. One retransmission packet per iteration is transmitted. This is the case for our experiment in ETS labs where flow rate of fresh packets was chosen to be between 1 and 2Mbps while retransmission packets were sent at 10 Mbps rate (10 BASE-T Ethernet link used between the two PCs and Network Impairment Generator). This was done to minimize the delay for recovery of lost Packets.

A six element integer array is dedicated for the generated packet. First element indicates whether the packet is fresh or retransmit (0 or 1) second element is the flow number (1 to 4). Third element is the generation iteration number while fourth and fifth elements are the block (window) and sequence numbers respectively. The last element is reserved to store iteration no at which the packet is delivered to the receiver.

3.1.2 Router

It is assumed that all links in the network have the same transmission capacity and all routers can receive one packet/iteration on each input port and transmit one packet/iteration on each output port.

For each output port in a router two FIFO (first in first out) priority queues [37] are implemented. The sender marks each packet with a priority. A fresh packet is marked with priority 0 and a retransmission packet with priority 1. The router always transmits packets out of the higher-priority queue (1) before moving on to the lower priority queue (0). Within each priority, packets are still managed in a FIFO manner. In fact, with above idea, retransmission packets (high priority) are put to the front of the line at each output port. Giving priority to retransmission packets is to reduce both queuing delay and packet loss due to buffer overflow (congestion).

In addition to queues, following elements may be implemented within a router depending on number of input and output links and also number of flows through them.

3.1.2.1 Packet Copier

When a packet is supposed to be transmitted to more than one branch of the multicast tree, a packet copier prepares required copies of each incoming packet and passes the copies to relevant buffers in the router. For example the packet copier implemented in router Rt7 makes six copies of incoming packets from flow no 2 and passes them to output buffer queues for transmission towards six receivers of that flow.

3.1.2.2 Mixer

When two different flows are input to two different input ports and are supposed to be output on the same output port, a mixer is used. The mixer is a buffer with a size of two and in each iteration, it may include 0, 1 or 2 packets depending on packets arrived at its

inputs. The packets in the mixer are passed to output buffer queues for transmission. As an example, the mixer in router Rt4 mixes flows no 2 and no 3 (Figure 3.1).

3.1.2.3 Separator

Packets of two mixed incoming flows, on the same link, are separated then packets of separated flows are forwarded to two separate output links. For instance, separator in router Rt2 separates flows no 2 and 3 and relevant packets are delivered to R 22 and R32. See Figure 3.1.

3.1.2.4 Background Traffic

For simulating background traffic (e.g. unicast, multicast other than the four considered flows and broadcast) random traffic is generated at each output buffer in a router with a certain probability P_b and this packet if generated will be sent over the output link, thus packets in the queue will be delayed or will be dropped if a fresh packet is input to a full buffer for that iteration. The procedure to decide whether a background packet is generated is as follows. A uniform distribution function (0,1) is called, if the output of the function is less than the generation probability P_b background packet is generated, otherwise it is not generated.

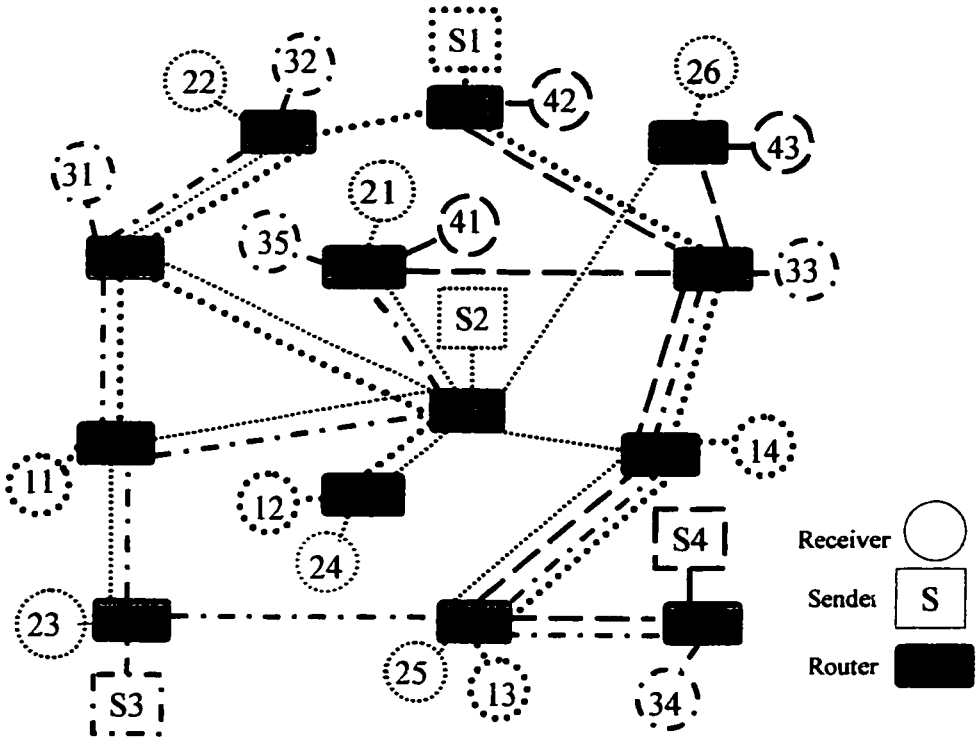
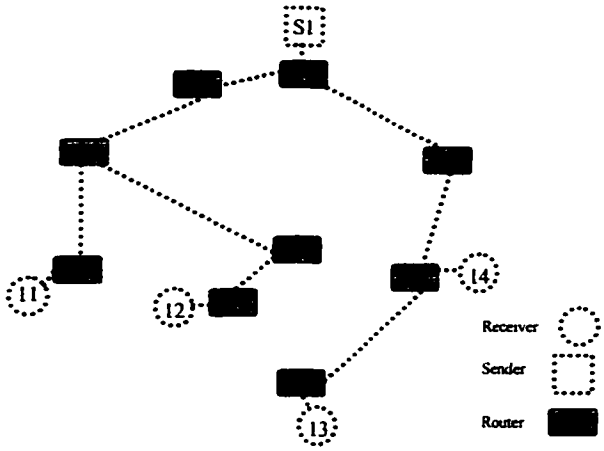
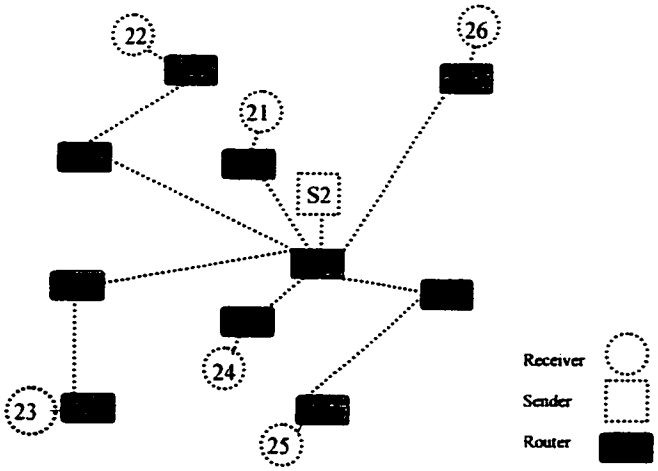


Figure 3.1: Hypothetical Multicast Network Topology

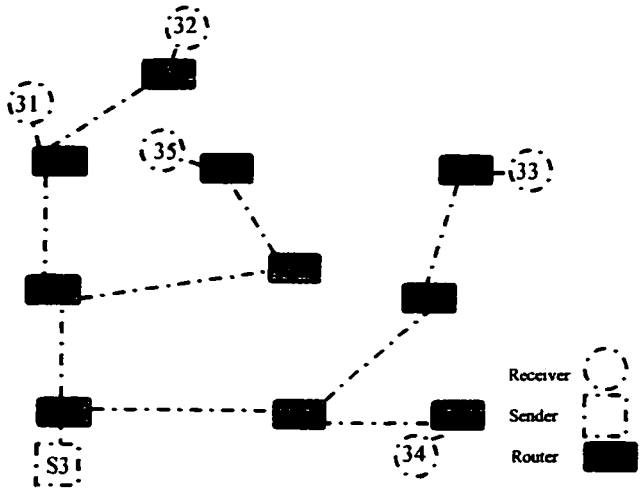


a. Flow 1 multicast tree

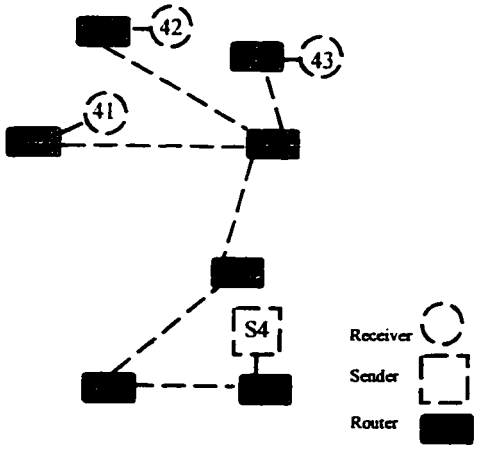


b. Flow 2 multicast tree

Figure 3.2: Separate trees of flows – continued in next page



c. Flow 3 multicast tree



d. Flow 4 multicast tree

Figure 3.2: Separate trees of flows – continued from previous page

flow 1 multicast routes
S1-Rt1-Rt2-Rt4-Rt8-R1(1)
S1-Rt1-Rt2-Rt4-Rt7-Rt10-R1(2)
S1-Rt1-Rt6-Rt9-Rt11-R1(3)
S1-Rt1-Rt6-Rt9-R1(4)

flow 2 multicast routes
S2-Rt7-Rt5-R2(1)
S2-Rt7-Rt4-Rt2-R2(2)
S2-Rt7-Rt8-Rt13-R2(3)
S2-Rt7-Rt10-R2(4)
S2-Rt7-Rt9-Rt11-R2(5)
S2-Rt7-Rt3-R2(6)

flow 3 multicast routes
S3-Rt13-Rt8-Rt4-R3(1)
S3-Rt13-Rt8-Rt4-Rt2-R3(2)
S3-Rt13-Rt11-Rt12-R3(4)
S3-Rt13-Rt11-Rt9-Rt6-R3(3)
S3-Rt13-Rt8-Rt7-Rt5-R3(5)

flow 4 multicast routes
S4-Rt12-Rt11-Rt9-Rt6-Rt5-R4(1)
S4-Rt12-Rt11-Rt9-Rt6-Rt1-R4(2)
S4-Rt12-Rt11-Rt9-Rt6-Rt3-R4(3)

Table 3.1: The Four Multicast Flow Routes

Flow No.	Next Hop	Flow No.	Next Hop
R t1		R t8	
1	Rt2	1	R11
1	Rt6	2	Rt13
4	R42	3	Rt4
		3	Rt7
R t2		R t9	
1	Rt4	1	R14
2	R22	1	Rt11
3	R32	2	Rt11
R t3		3	Rt6
4	R43	4	Rt6
2	R26	R t10	
R t4		1	R12
3	Rt2	2	R24
3	R31	R t11	
1	Rt8	1	R13
1	Rt7	2	R25
2	Rt2	3	Rt9
R t5		3	Rt12
2	R21	4	Rt9
3	R35	R t12	
4	R41	3	R34
R t6		4	Rt11
4	Rt1	R t7	
4	Rt3	1	Rt10
4	Rt5	2	Rt3
1	Rt9	2	Rt9
3	R33	2	Rt10
R t13		2	Rt8
2	R23	2	Rt4
3	Rt8	2	Rt5
3	Rt11	3	Rt5

Table 3.2: Routing Tables for the Four Multicast Flows

3.1.3 Receiver

As stated above each block of information contains n encoded packets that are transmitted one after another by the sender and distributed by the network to the group members (receivers) of the flow. Time taken to generate n packets is WL (Window Length)=n*gR iterations. For the case of gR=4 and n=64,WL=256 i.e. it will take 256 iterations to generate a block of encoded packets of data and repair. Upon receiving first packet of the flow, the receiver determines and sets first window timer. The time allowed for receiving packets of the window is WL+ND (Network Delay). Receiver will continue receiving and storing packets, making note of number of lost packets and also their block and sequence numbers until one of the three followings happens:

1. Timer for receiving packets of data and repair expires (Hereafter data and repair packets that are transmitted for the first time are called ***fresh*** while the packets retransmitted in response to NAKs are called ***retransmitted*** ones).
2. Last packet of the window is received. This means that no more fresh packets of this window will arrive.
3. A packet belonging to next window is received. This is again an indication that no more packets from the current window will arrive.

In all three above cases, receiver checks the number of received packets (p) of the window. Three cases may happen:

1. If p is greater than or equal to k=56 and none of the data packets (Sequence Number 1 to k) are missing. No decoding is required. Since the RS code for encoding is systematic the first k packets are data and in order and can be stored in output buffer for playing. The window number is incremented and receiver starts receiving packets of the new window.
2. If p is greater than or equal to k=56 and all data packets are not received (i.e. one or more of packet numbers 1 to k are missing). Decoding is required. Decoding delay will be added to other delays. After decoding next window will start as stated above.
3. If p< k the decoder can't decode and recover the lost packets. As a result first NAK for q=k-p missing packets containing their block and Seq. Number is transmitted to the sender. A timer for first request is started. If q packets are

received, before the timer is expired, the decoder will decode and missing packets are recovered. Then processing of next window starts. In case the timer is expired and all requested packets are not received 2nd NAK will be sent. This process can continue until all packets are recovered. In our case maximum number of NAKs are limited to 2.

Figure 3.3.a and b show flow chart of the code for FEC/ARQ receiver used in the simulation.

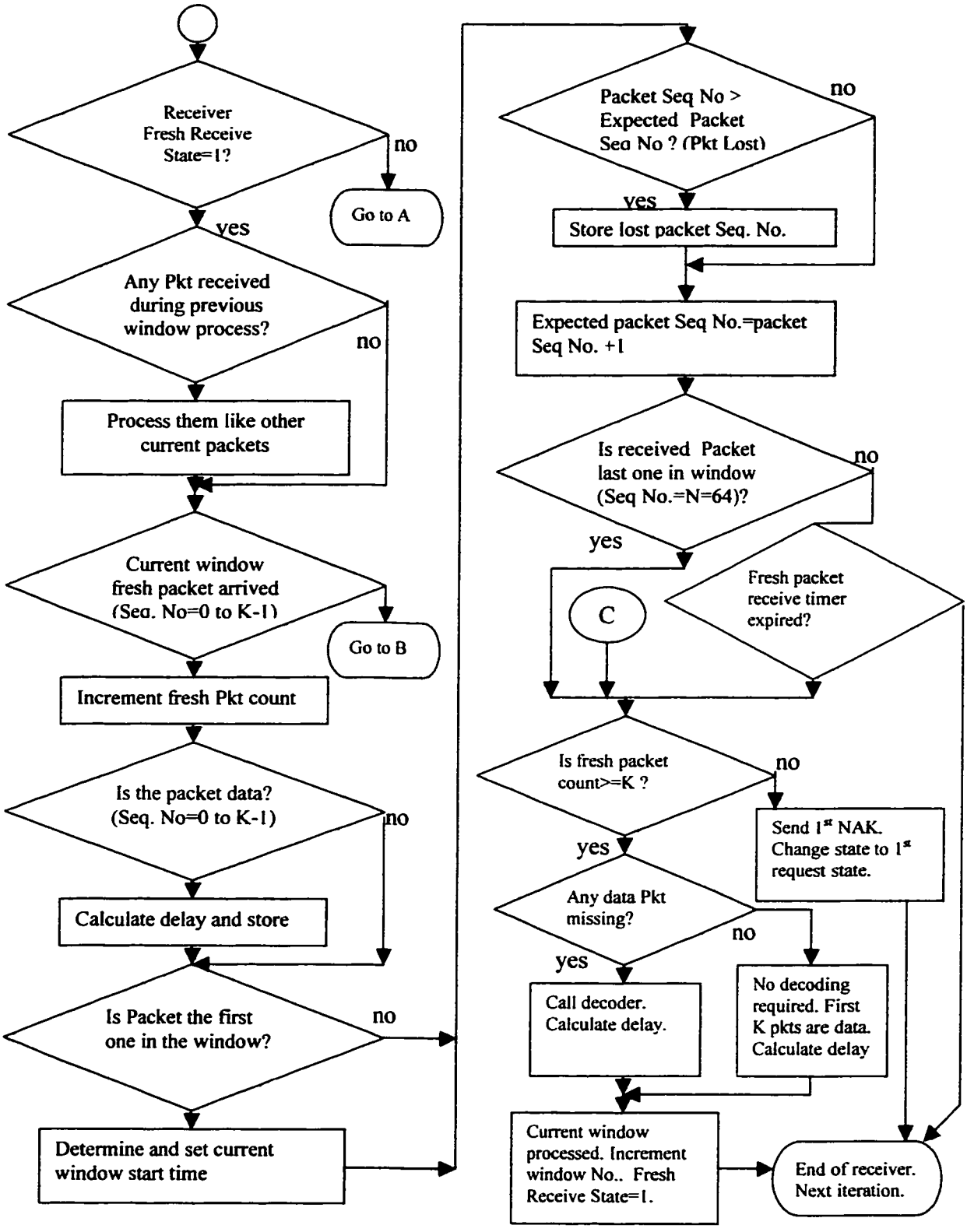


Figure 3.3: FEC/ARQ Receiver flow chart used in simulation, continued in next page

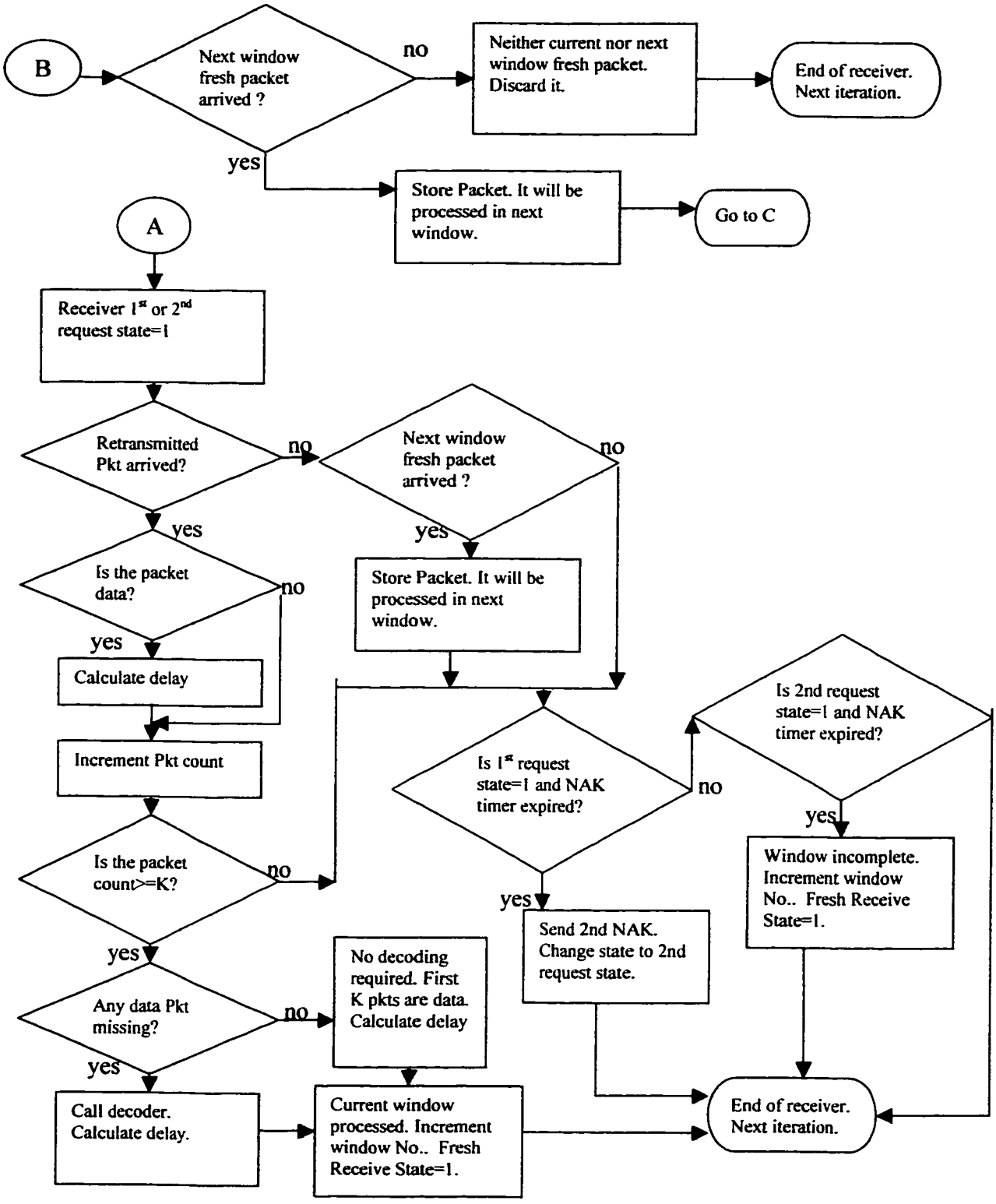


Figure 3.3: FEC/ARQ Receiver flow chart used in simulation, continued from previous page.

3.2 ARQ Scheme

We simulate a generic version of a reliable multicast protocol that uses only ARQ to recover the lost packets. The results of this simulation will be compared to the hybrid FEC/ARQ protocol so that the performance of the latter can be evaluated correctly. The receiver in ARQ scheme will be different from FEC/ARQ and the sender will have a minor change. Other components of the hypothetical multicast network such as routers and links are the same for both protocols.

3.2.1 Sender

The sender starts and continues to send Fresh (data) packets until it receives a negative acknowledgement (NAK) from a receiver for a missing packet. The sender will stop sending Fresh packets and will retransmit the requested packet. After retransmission the sender will continue sending Fresh packets again.

3.2.2 Receiver

Receiver starts to receive packets sent by the sender. When it detects a missing packet it sends a NAK for that packet to the sender and will start a timer. If the timer expires before the requested packet arrives, the receiver will send another NAK and start a new timer. This process can continue until the lost packet is received. In our simulation we have limited the number of NAKs to two so that both protocols can be run and compared at the same conditions.

3.3 NAK Implosion Avoidance

As we shall see later in detail, one of the advantages of FEC/ARQ scheme is that it solves the problem of NAK implosion to a great extent because for packet losses up to several percents there is no need for NAKs. So more processing and complexity at designated routers, receivers and sender for handling NAK implosion problem is avoided. In our simulations we compare behavior of the two multicast protocols when they are equipped with NAK implosion avoidance [38]. Brief description of NAK implosion avoidance operating mechanism was given in section 1.6.1.2.

3.4 Simulation Results

Throughout this section, performance of integrated FEC/ARQ reliable multicast protocol is investigated and where necessary, the results are compared to the scheme with only ARQ. First we assume that packet losses occur independently. Then we examine the effect that different loss probabilities have on the performance of reliable multicast protocol. Finally, performance of hypothetical multicast network is studied, while the routers and their buffers are considered as sources of packet loss (buffer overflow) and network delay (buffer delay).

3.4.1 Independent Loss - Homogeneous Receivers

In this case it is assumed that only the receivers lose packets independently with probability p. Other nodes (routers) of the multicast tree do not lose packets at all. Figure 3.4.a shows the number of transmissions versus packet loss for the four flows and results of FEC/ARQ scheme is compared to the case without FEC (only ARQ). No of transmissions is normalized to number of data packets sent. So for ARQ protocol the number of transmissions at no loss is equal to number of transmitted data Packets i.e. 1, while number of transmissions for FEC/ARQ at no loss is N/K= 1.14, that is due to sending parity as well as data. As seen in the figure, while in ARQ only, increasing number of receivers in a flow increases the number of transmissions considerably, for FEC/ARQ schemes it causes a minor increase at high loss rates (around 10 %) and the rate remains constant and almost equal to 0 at low packet loss probabilities (due to action of FEC). As a result while ARQ schemes suffer from weakness in scalability, FEC/ARQ protocol is scalable and can be used with large number of receivers.

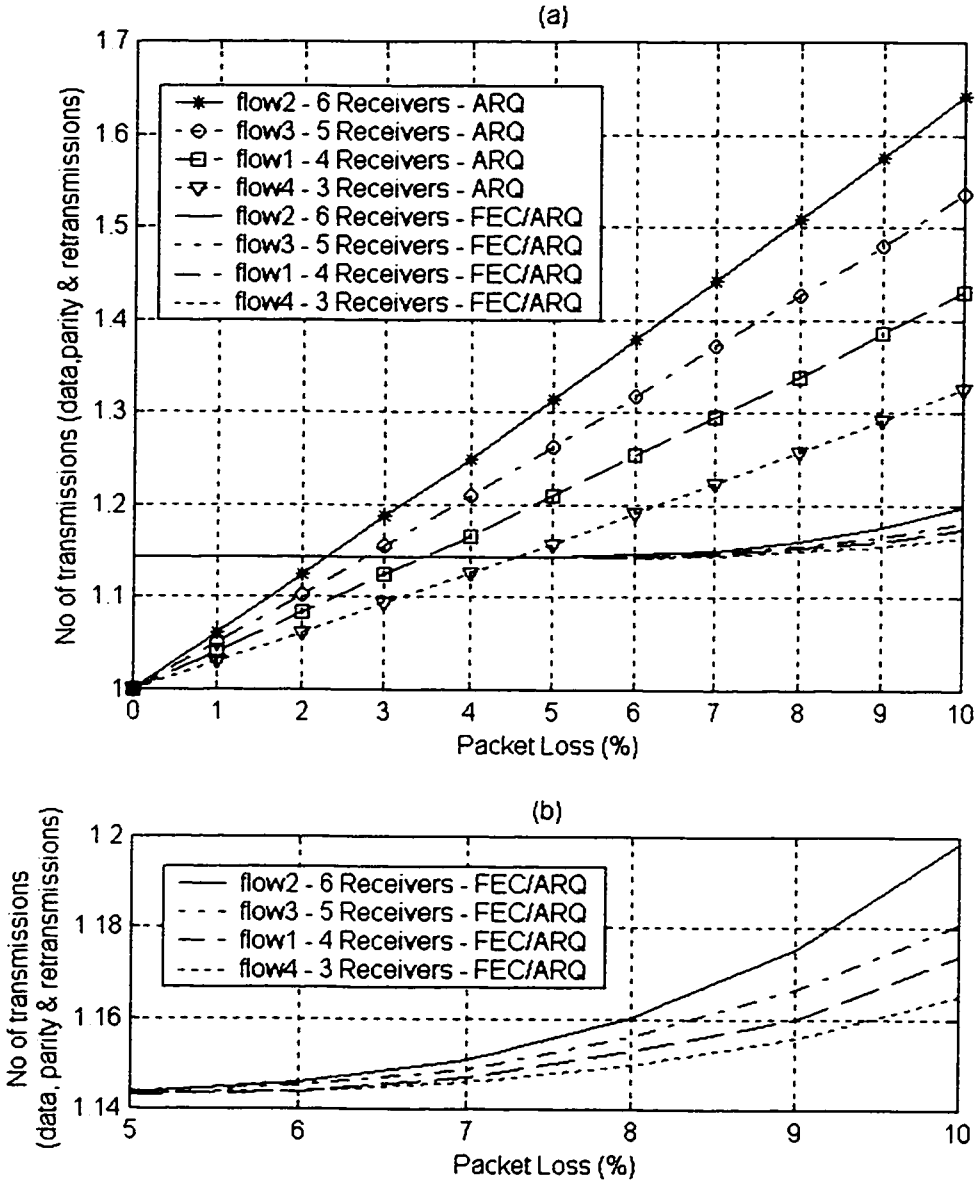


Figure 3.4: a) Number of Transmissions versus Packet Loss. Independent Packet loss, Homogeneous Receivers, FEC/ARQ and ARQ schemes. b) FEC/ARQ curves of figure (a) are enlarged for clarity.

Figure 3.5 compares the two protocols when NAK implosion avoidance capabilities in network elements are considered for ARQ scheme. Although number of retransmissions for ARQ compared to previous figure is decreased, this protocol can't scale well yet. Figure 3.6 compares number of retransmissions (as a percentage of data packets transmitted) for ARQ with NAK implosion avoidance and FEC/ARQ schemes. Again very low percentage of retransmissions in FEC/ARQ is a measure of efficient bandwidth utilization and scalability of this protocol.

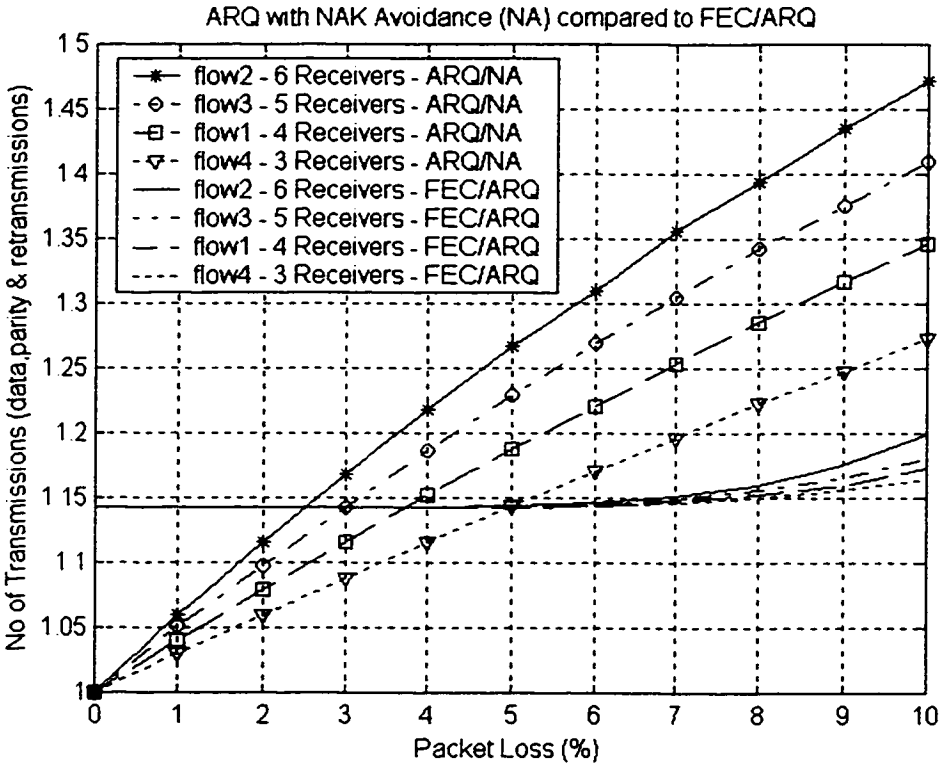


Figure 3.5: Number of Transmissions versus Packet Loss. Independent Packet Loss, Homogeneous Receivers, ARQ with NAK Avoidance and FEC/ARQ schemes.

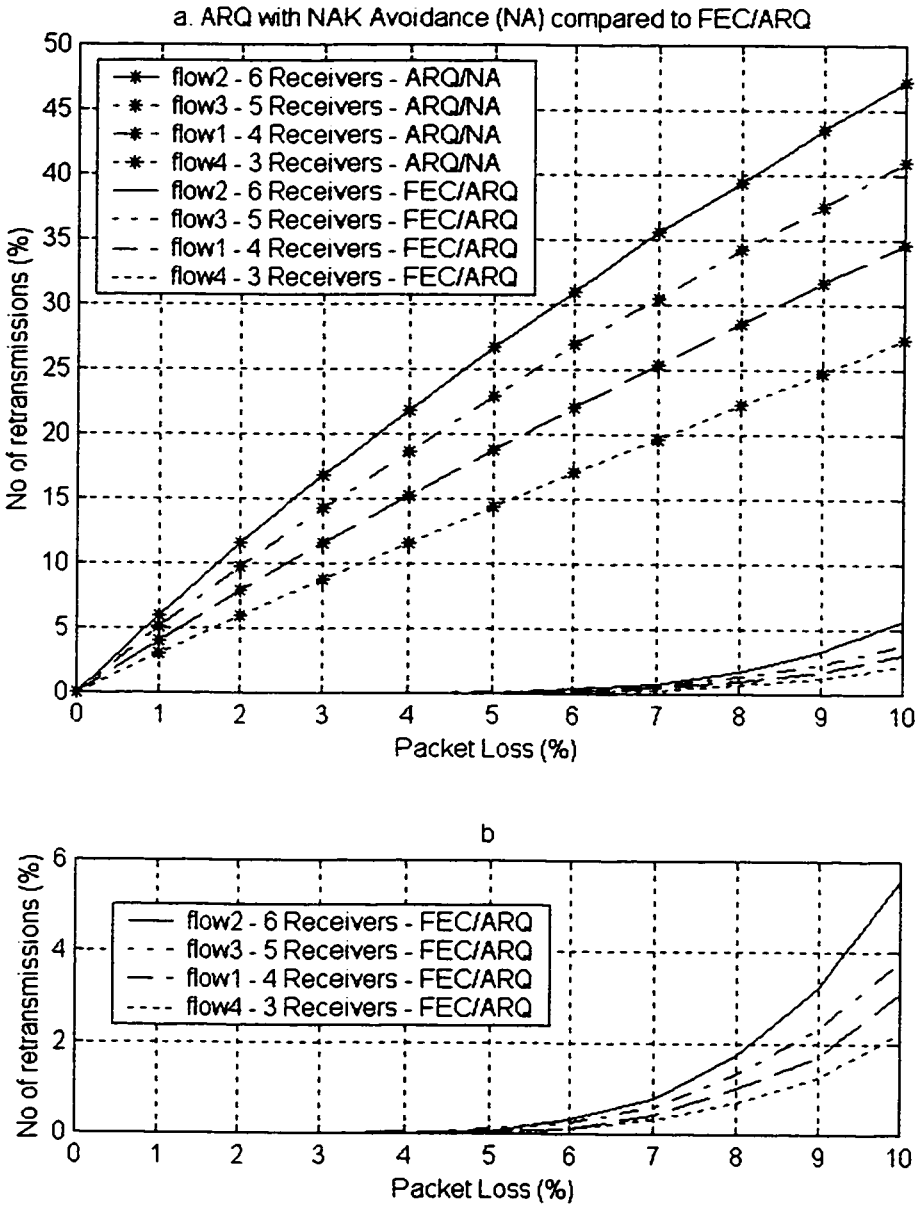


Figure 3.6: a) Number of Retransmissions versus Packet Loss. Independent Loss, Homogeneous Receivers, ARQ with NAK Avoidance and FEC/ARQ schemes. b) FEC/ARQ curves of figure (a) are enlarged for clarity.

In Figure 3.7.a, number of NAKs as a percentage of number of windows is plotted versus packet loss. It should be mentioned that in the FEC/ARQ protocol, each receiver sends NAKs at the end of each block and the NAK includes sequence number of required packets from that window. For instance, at 10% packet loss for flow no.1 with 4 receivers, for almost every window a NAK is transmitted (windows generated NAK equals 100%). Figure 3.7.b and c show Number of first and second sent NAKs. The values for number of retransmissions for the two former figures are added together to achieve Figure 3.7.a.

Our simulation shows that for ARQ scheme, number of NAKs increases almost linearly with packet loss and also with the number of receivers of a flow (Figure 3.8). Comparing this figure to Figure 3.7, it is observed that for integrated FEC, number of NAKs is considerably smaller than the case without FEC.

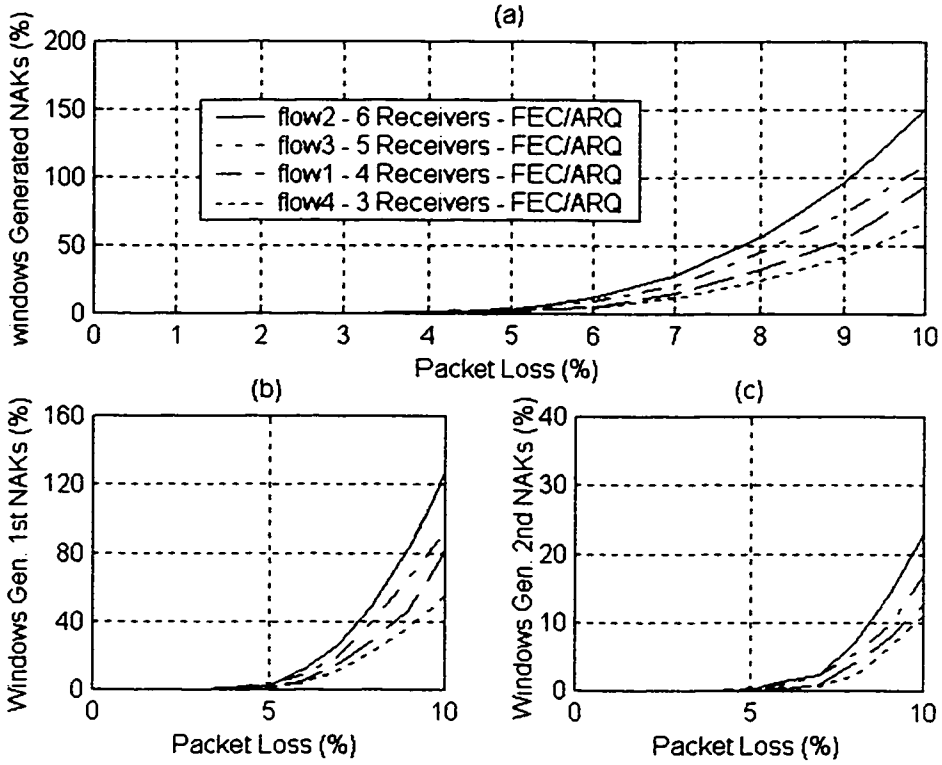


Figure 3.7: a) Windows Generated NAKs versus Packet Loss. b) Windows Generated 1st NAKs. C) Windows Generated 2nd NAKs. Independent Packet Loss, Homogeneous Receivers, FEC/ARQ scheme.

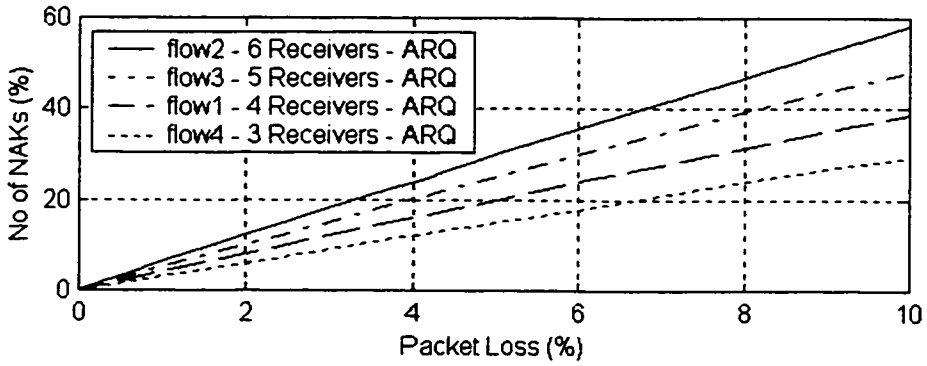


Figure 3.8: Number of NAKs versus Packet Loss. Independent Packet Loss, Homogeneous Receivers, ARQ scheme.

In Figure 3.9.a, Residual Loss (RL) in percent is plotted versus packet loss. Residual loss (%) is calculated according to following formula,

$$RL = 100 * \frac{DNR}{DL} \tag{3.1}$$

in which DNR (Data Not Recovered) is the number of lost data packets of the flow that are not recovered and DL is the total number of lost data packets of the flow. As is seen, FEC/ARQ scheme is almost reliable (less than 0.1% for losses up to 10 percent and completely reliable for losses below 5%, with max number of NAKs/window limited to two). Residual loss for ARQ with the same number of allowed NAKs/lost packet is at least 10 times the value for FEC/ARQ (see Figure 3.9.b). In practice for ARQ protocol, when reliable communication is required, greater number of retransmissions for each lost packet is allowed.

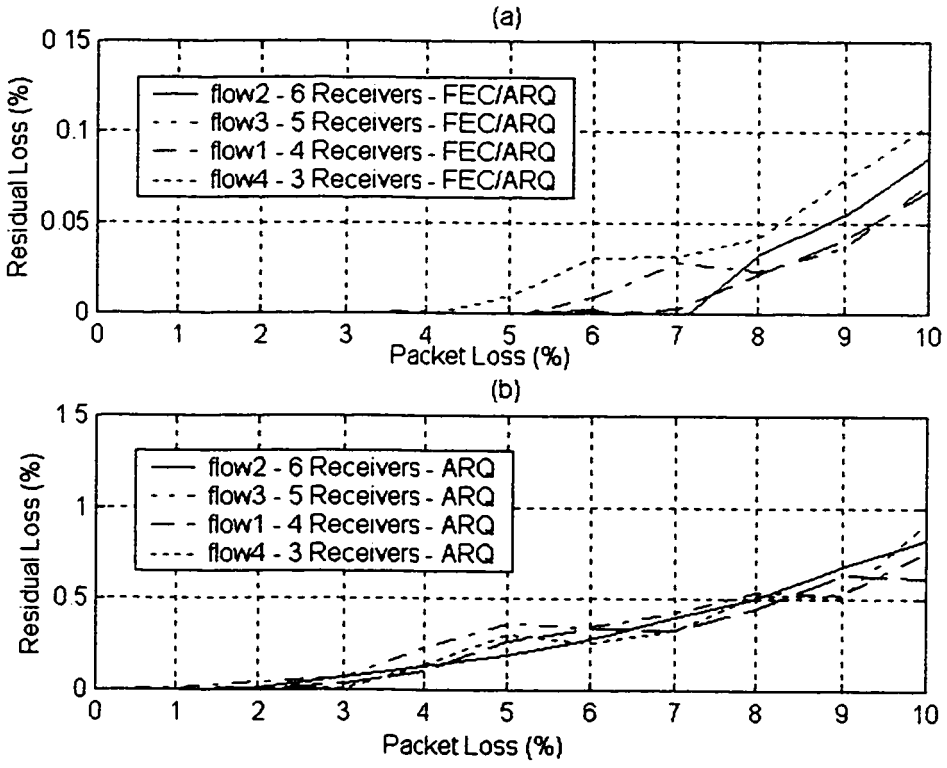


Figure 3.9: Residual Loss versus Packet Loss. Independent Packet Loss, Homogeneous Receivers. a) FEC/ARQ. b) ARQ.

Packet delay is also measured as the time difference between the time the packet is generated and the time it is ready to be played at the receiver (equation 2.3). In this section, it is assumed that transmission delays (i.e. encoding and interleaving,..) are small compared to receiving delays and thus they are ignored. While the packet is transmitted in the network it is delayed for one unit of time (iteration) whenever it passes a node. This is the only network delay that is considered. Queuing delays are considered in the last section of this chapter (3.4.4) where routers and their buffers act as sources of packet loss and queuing delays in the hypothetical multicast network. The delay measured in this section also includes all processing delays (receiving, decoding, de-interleaving, transmitting NAKs and receiving retransmissions). Using equation 2.4 and applying relevant subscripts we have:

$$E(i,j) = \sum_{k=1}^{N_P(i)} \frac{\Delta_k(i,j)}{N_P(i)} \tag{3.2}$$

where $E(i,j)$ is the receiver average (mean) delay, $N_p(i)$ is the number of packets of flow i, $\Delta_k(i,j)$ is the delay of packet number k, i is the flow no. (sender no.) and j is the receiver number. Flow average delay $E(i)$ is the mean of all receivers' average delay and is given below:

$$E(i) = \frac{\sum_{j=1}^{N_R(i)} E(i,j)}{N_R(i)} \tag{3.3}$$

where $N_R(i)$ is the number of receivers of the flow i.

Replacing notations of equation 3.3 and 3.4 in equation 2.5 we get the following equation for variance of flow delay ($\sigma_{E_i}^2$),

$$\sigma_{E_i}^2 = \frac{\sum_{j=1}^{N_R(i)} \sum_{k=1}^{N_P(i)} \left(\Delta_k(i,j) - E(i)\right)^2}{N_R(i) * N_P(i) - 1} \tag{3.4}$$

Figure 3.10.a is the plot of average delay of four flows of the multicast network. Relevant curves of delay variance are plotted in Figure 3.10.b. Peak delay is shown in Figure 3.11. All delays are measured and calculated at 2 Mbps transmission rate. The reasoning for large values of delay variance in Figure 3.10.b is similar to explanation given in 2.5.2.1 for our experimental results.

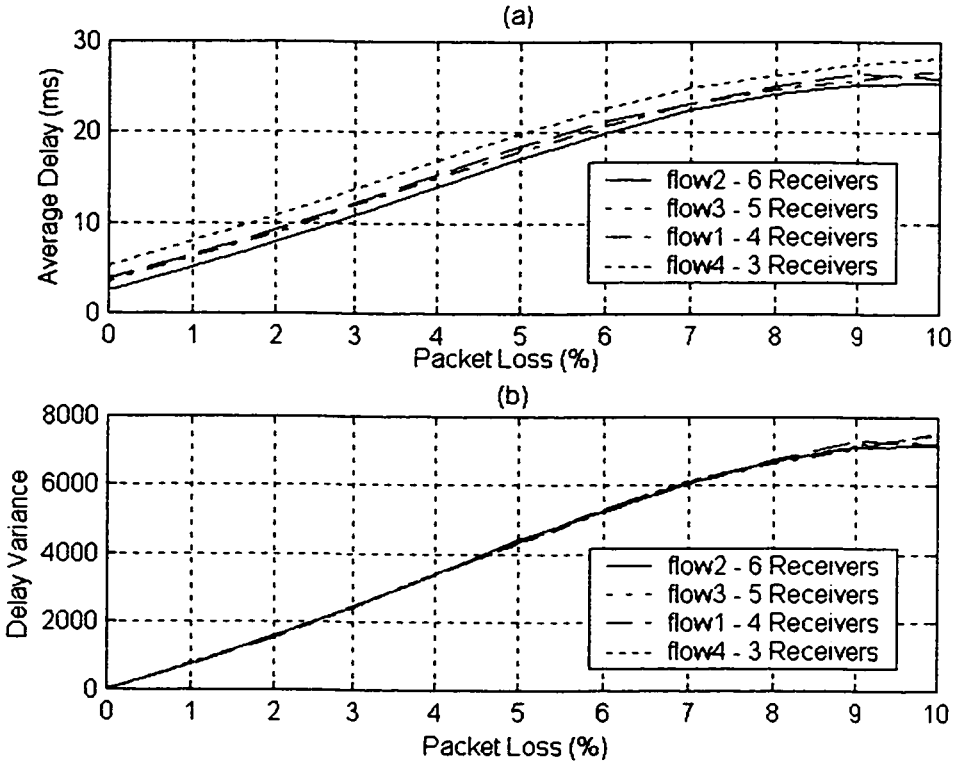


Figure 3.10: a) Average Delay versus Packet Loss. b) Delay Variance versus Packet Loss. Independent Packet Loss, Homogeneous Receivers, FEC/ARQ scheme.

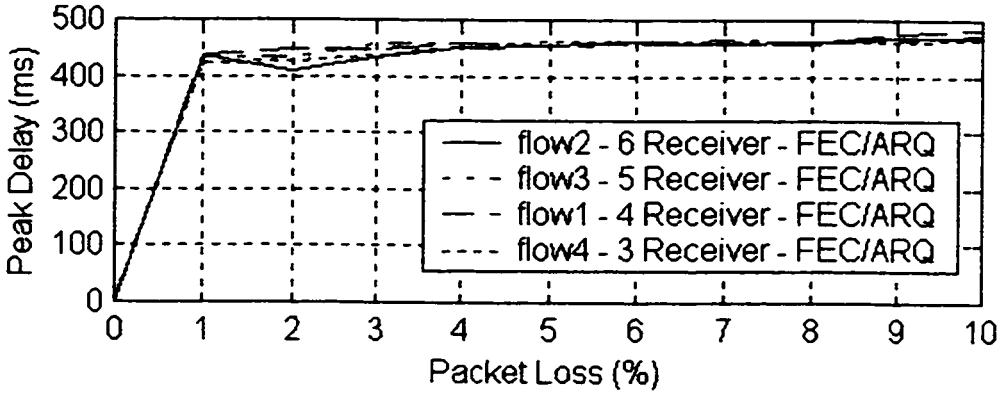


Figure 3.11: Peak Delay versus Packet Loss. Independent Packet Loss, Homogeneous Receivers, FEC/ARQ scheme

3.4.2 Shared loss – Homogeneous Receivers

In previous section we have assumed that all losses occur as independent events at the receivers while in reality there may be losses within the multicast tree that will be shared by more than one receiver. In this section we investigate whether and how the presence of such losses affects the conclusions drawn from our independent loss model. Instead of using our hypothetical multicast network, we use another model to simulate shared losses. In [39] the sharing of loss was analyzed for multicast trees built by different multicast routing algorithms. The authors conclude that the loss sharing in multicast trees is modeled well by a full binary tree (FBT). In order to investigate the effect of shared loss we consider a FBT of height d, where the sender is the root of the tree and the receivers are the leaves. We assume that losses occur as independent events at each node (router) excluding source (sender) and leaves (receivers), with probability p_n. Here p_n is chosen such that loss at each receiver is p,

$$p = 1 - (1 - p_n)^{(d+1)} \tag{3.5}$$

We consider three full binary trees of 2, 4 and 8 receivers (values of 0, 1 and 2 for d respectively) and execute simulations for them.

Independent Loss: In order to be able to compare results of independent and shared loss cases, the same full binary trees as for shared loss are used to simulate independent losses and it is assumed that in this case only the receivers lose packets, each receiver independently with probability p. Other nodes of the multicast tree do not lose packets at all.

Note that each receiver experiences the same loss probability p in both shared and independent models.

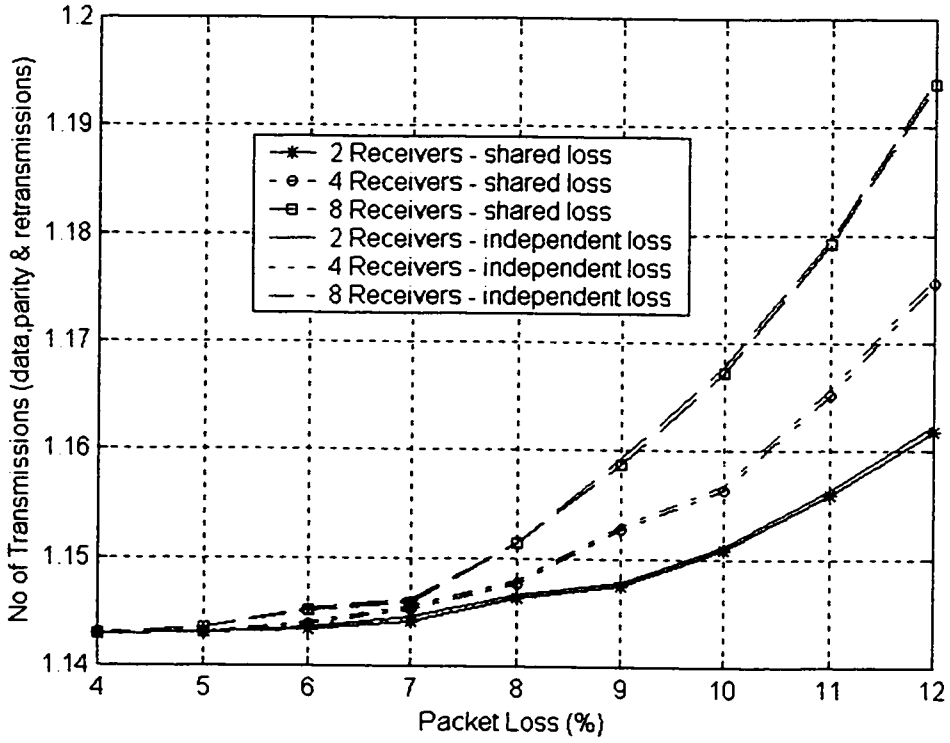


Figure 3.12: Comparison of No of Transmissions versus Packet Loss for Independent and Shared losses. Homogeneous Receivers. FEC/ARQ scheme.

Figure 3.12 compares the number of transmissions for independent and shared losses. First, we observe that the number of transmissions is almost the same for both types of loss when NAK avoidance is not considered. Second, it is seen that our observations drawn from the independent loss case in the previous section continue to hold. However, comparing Figure 3.13 and Figure 3.14 it is seen that the number of receivers for shared losses need to be larger before the benefits of integrated FEC appears. These figures show that the overhead of transmitted parities with integrated FEC is amortized by the repair efficiency for greater number of receivers for shared losses than for independent losses.

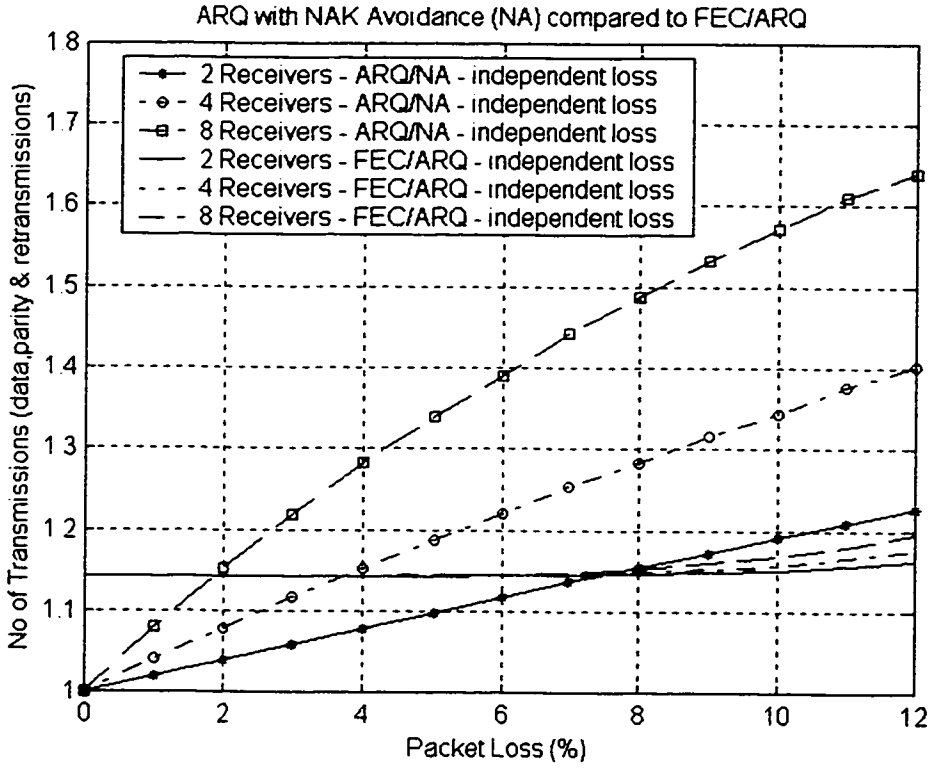


Figure 3.13: No of Transmissions versus Packet Loss. Independent Packet Loss, Homogeneous Receivers, ARQ with NAK Avoidance and FEC/ARQ schemes.

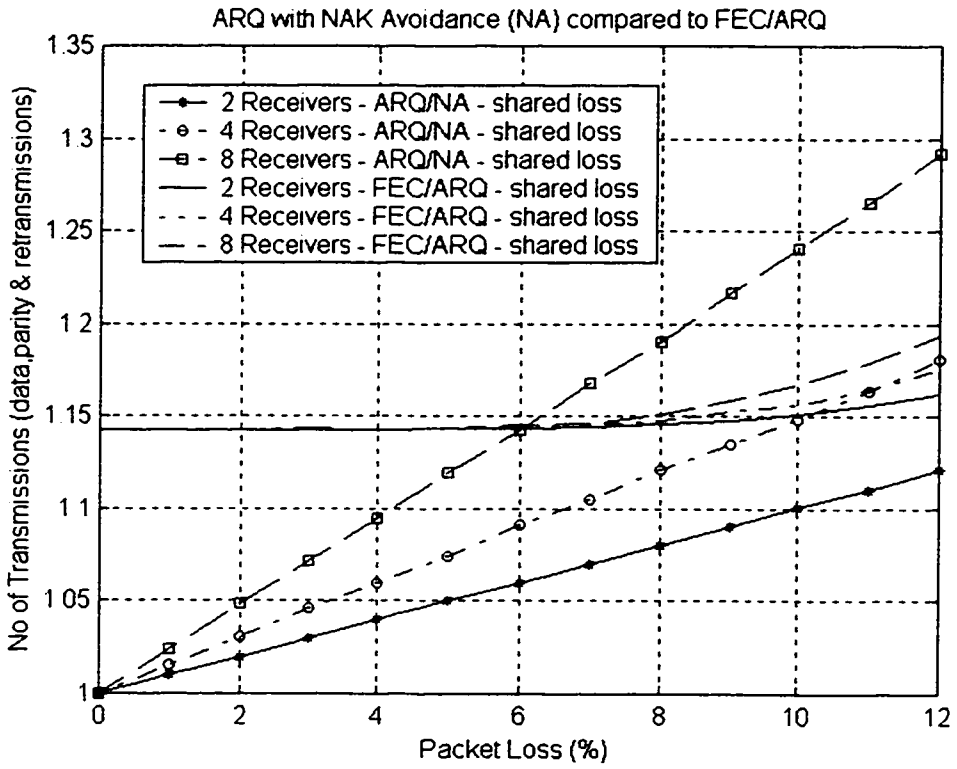


Figure 3.14: No of Transmissions versus Packet Loss. Shared Packet Loss, Homogeneous Receivers, ARQ with NAK Avoidance and FEC/ARQ schemes.

3.4.3 Heterogeneous Receivers

In this section we are going to discuss how our observations change in the presence of heterogeneous receivers, i.e. receivers with different loss probabilities. We use our Full Binary Tree (FBT) model for 2, 4 and 8 receivers as in the shared loss case. We assume that only receivers lose packets and half of the receivers are low loss and the other half are high loss. High loss receivers' loss is assumed to be 9 times low loss receivers'.
First, we compare performance of FEC/ARQ with the case without FEC (only ARQ) and consider the effect of NAK avoidance mechanism on the performance. Then behavior of heterogeneous receivers is compared to homogeneous receivers. Note that for comparison, for homogeneous receivers we have used our Full Binary Tree (FBT) model for 2, 4 and 8 receivers as in the heterogeneous case.

Figure 3.15 shows the number of transmissions of heterogeneous receivers for both FEC/ARQ and only ARQ schemes. Again it is seen that FEC/ARQ is more advantageous but comparing those graphs with Figure 3.13 (for which receivers are homogeneous i.e. there are no high loss receivers) shows the degradation due to high loss receivers. This means that number of receivers has to be higher before advantages of integrated FEC reveals. Also comparing Figure 3.15 and Figure 3.13 we observe that the presence of high loss receivers has a greater effect on the number of transmissions in the case of integrated FEC than only ARQ. Simulations are also performed for the case without NAK Avoidance. Results are similar when NAK avoidance is not provided and it is observed that better performance for heterogeneous receivers by using FEC/ARQ is achieved.

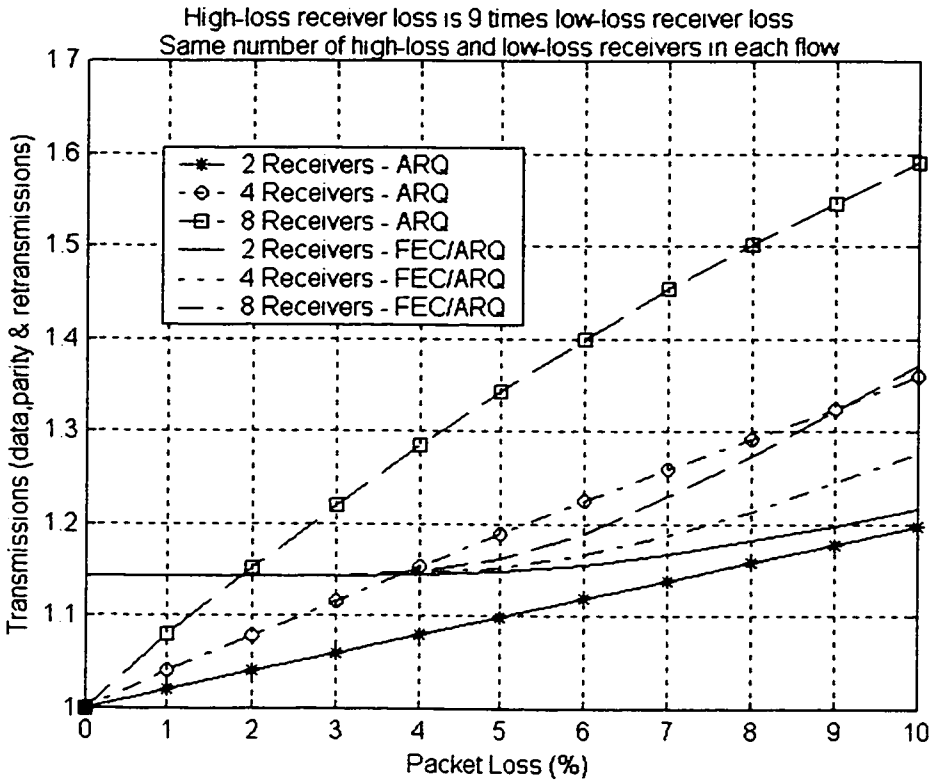


Figure 3.15: No of Transmissions versus Packet Loss. Heterogeneous Receivers, ARQ with NAK Avoidance and FEC/ARQ schemes. Number of High-loss and Low-loss receivers in each flow is the same.

Figure 3.16 compares number of transmissions for heterogeneous receivers with homogeneous receivers. It is observed that the performance of integrated FEC is determined by high loss receivers. For instance, in heterogeneous case with four receivers and at 5% average loss (i.e. 2 receivers with 9% loss and 2 receivers with 1% loss), number of transmissions is 1.153, that is equal to the value for the flow with two homogeneous receivers at 9% loss. As another example, it is seen that number of transmissions for heterogeneous model with 8 receivers at 6% loss (i.e. 4*10.8% and 4* 1.2% loss receivers) is 1.19 and it is equal to number of transmissions for homogeneous case with 4 receivers at 10.8% loss.

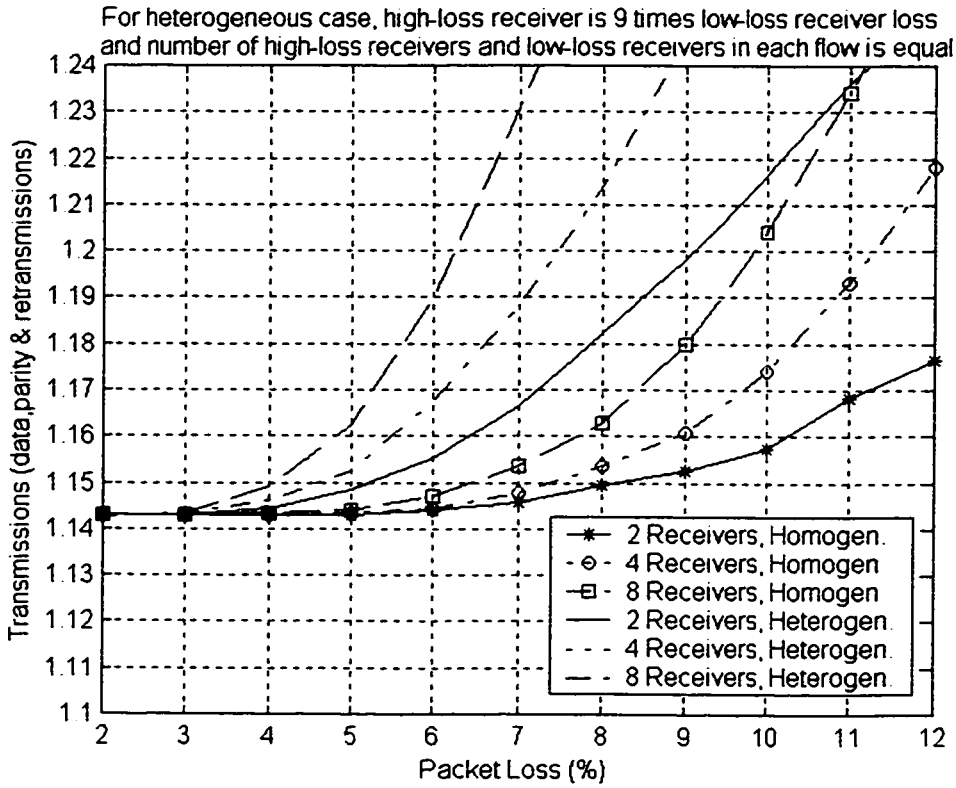


Figure 3.16: Comparison of no of transmissions between homogeneous and heterogeneous receivers. FEC/ARQ scheme.

3.4.4 Considering queuing delay and loss due to buffer overflow

In section 3.4.1 we studied the performance of our hypothetical multicast network for homogeneous receivers with independent packet loss. In those simulations, we assumed that while the packet was transmitted in the network it was delayed for one unit of time (iteration) whenever it passed a node. This was the only network delay that was considered.

In this section we also consider queuing delays in routers and assume that packet losses are due to router buffer overflow (network congestion). In order to understand how these assumptions may affect our conclusions, first simulation results and performance of a single link and then our hypothetical multicast network are studied. Buffer size of the routers in the network for the simulation is 7.

1. Single link: A single end-to-end link consisting of a sender, a router and a receiver and their interconnecting links is considered. Random background traffic is generated at input to the router as stated in section 3.1.2.4. Background traffic generation rate is changed and parameters like packet loss, delay and no of transmissions and etc. are measured and relevant graphs are plotted. Service rate of the router buffer i.e. the rate at which packets leave the router is assumed to be constant and it is one packet per iteration. Also main flow rate i.e. flow from sender to receiver, is 0.25 times the service rate of the buffer. Figure 3.18.a shows packet loss in (%) due to buffer overflow versus traffic intensity ($\rho = \lambda / \mu$ in packet arriving / packet served) of the router buffer. μ is packet serving rate, $\lambda = \lambda_1 + \lambda_2$ is the total packet arriving rate, λ_1 is main flow arriving rate and λ_2 is background traffic arrival rate at the buffer. In the above figure $\mu = 1$ and λ_1=0.25 packet/iteration are kept constant and background traffic (λ_2) is changed.

Figure 3.17.a and b show the plot of mean delay and delay standard deviation versus packet loss. The dashed curves are plotted without considering the router buffer and it is assumed that packets are randomly lost at the receiver whenever the output of a uniform distribution function after each call is less than the probability of required loss P_L. Also the delay (Δ_t as defined in eq. 2.1) does not include the network delay caused by queue in the buffer. For the solid curves the router is considered and packet losses are due to the router buffer overflow and end to end delay of the packets (Δ_t) also include delay caused by queue in the buffer. It is seen that average delay in solid graph is about 10 units of delay (iterations) more than in the dashed curve and this is due to queuing delay in the buffer. As shown in Figure 3.17.b standard deviation of the case with buffer is less than the other curve that is reasonable.

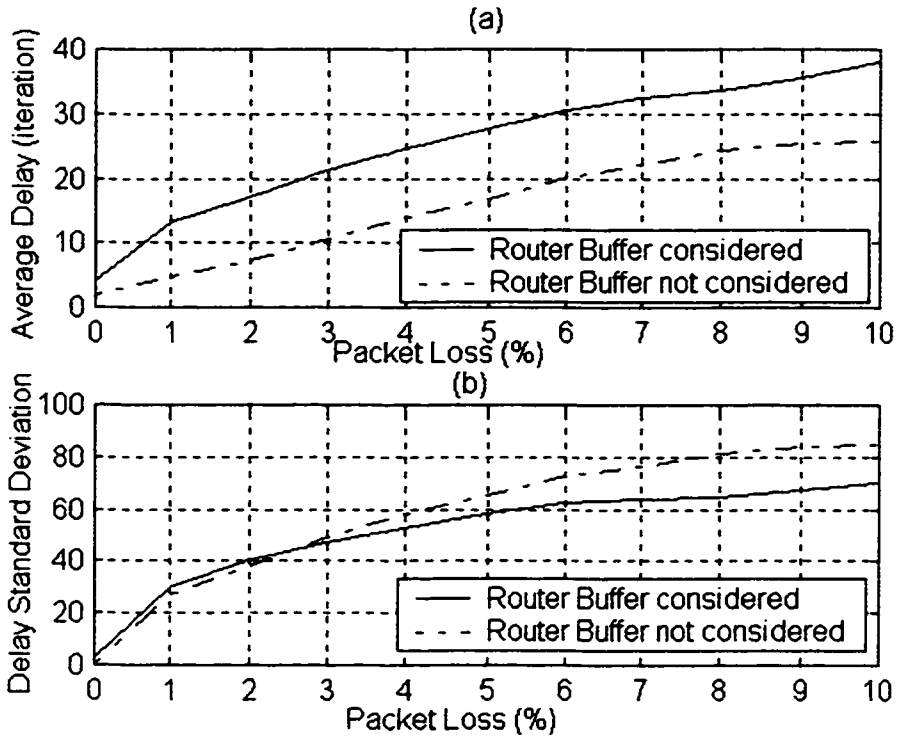


Figure 3.17: A single end to end link. Average Delay (a) and Delay Standard Deviation (b) versus Packet Loss.

In Figure 3.18.b peak delays are shown. Maximum peak delay for the case with router buffer is about 500 that is less than the maximum tolerable delay for FEC/ARQ scheme (520 ms at 2 Mbps rate, section 2.5.2.4). It means that none of the packets are considered lost due to delay in arriving at receiver. For longer delays in buffer (peak delays in excess of 520 ms) the packets are considered lost (late arrival at receiver and expiry of receive timer in receiver). In this case the solution is to transmit the packets by the sender at lower rates. In practical situations buffer delays in network routers are rarely as high as 20 ms and flow rates of about 2Mbps seem achievable. Figure 3.18.c shows no of transmissions versus packet loss due to buffer overflow. Note that values on vertical axis are normalized to K (no of the original data packets). As a result at zero packet loss, no of transmissions (data and parity) is 1.142 (N/K).

Above observations show that our previous conclusions continue to hold when queue in the router is considered for a single link.

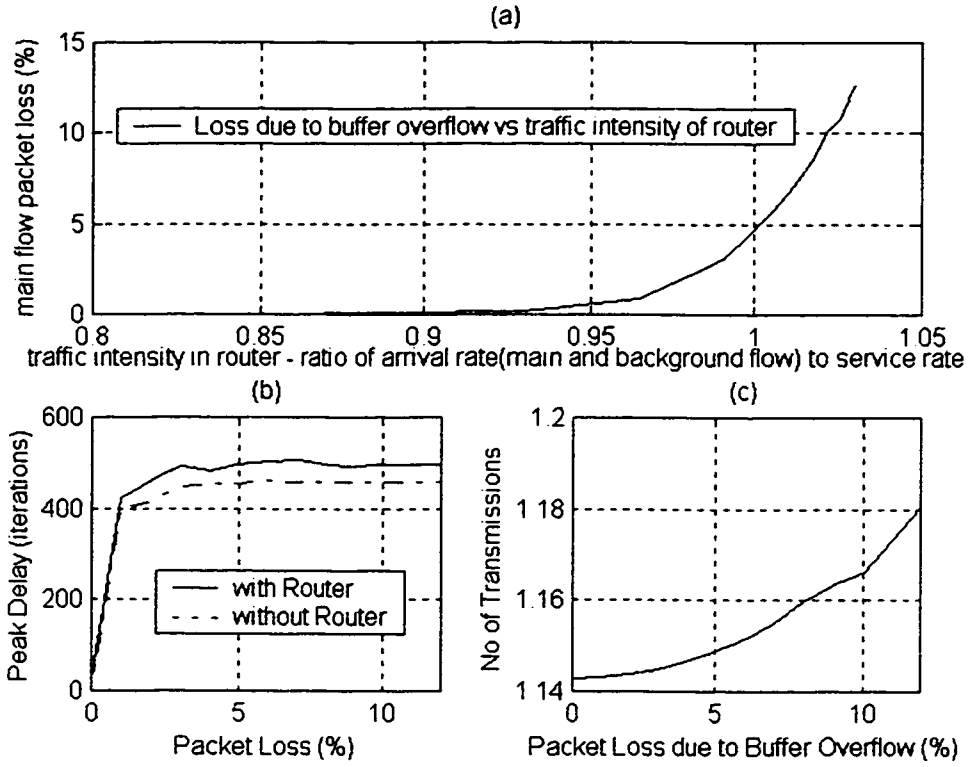


Figure 3.18: A single end-to-end link including one sender, one router and one receiver. FEC/ARQ scheme. a. Loss due to buffer overflow versus router traffic intensity, b. Peak delay versus packet loss and c. No of transmissions versus loss due to buffer overflow.

2. Hypothetical Multicast Network: Background traffic is generated at the input to each router buffer as in single link above. Main flow is also input to each router buffer and the rate is assumed to be constant and equal to 0.25 times service rate of buffer. Depending on network topology, packets from other three multicast flows of the hypothetical network may be input to the buffer, too. As a result, buffers in our network have different traffic intensities at the same background traffic that is the reason for different queuing delays in buffers and also losses due to buffer over flow. For instance, for flow no 3, on the link from the sender (S3) to R31, there are only two buffers of routers RT13 and Rt8 and only background and flow number two traffic are input to

them. While for all flow number 4 links, e.g. link from S4 to R41, there are four buffers (routers Rt12, Rt11, Rt9 and Rt6) two of which (Rt11 and Rt9) also carry the packets from flow no 3 and as a result will have high loss and high delay at the same background traffic (Figure 3.1). Since all three links of flow no 4 are of highest loss and delay, this flow has maximum average loss and delay compared to other three flows and this is clearly seen in next figures. Figure 3.19.a and b show average delay E(i) including buffer delay and packet loss versus background traffic rate respectively. While drawing the figures, main flow and other multicast flow rates input to buffer are kept constant and equal to 0.25% of buffer output rate.

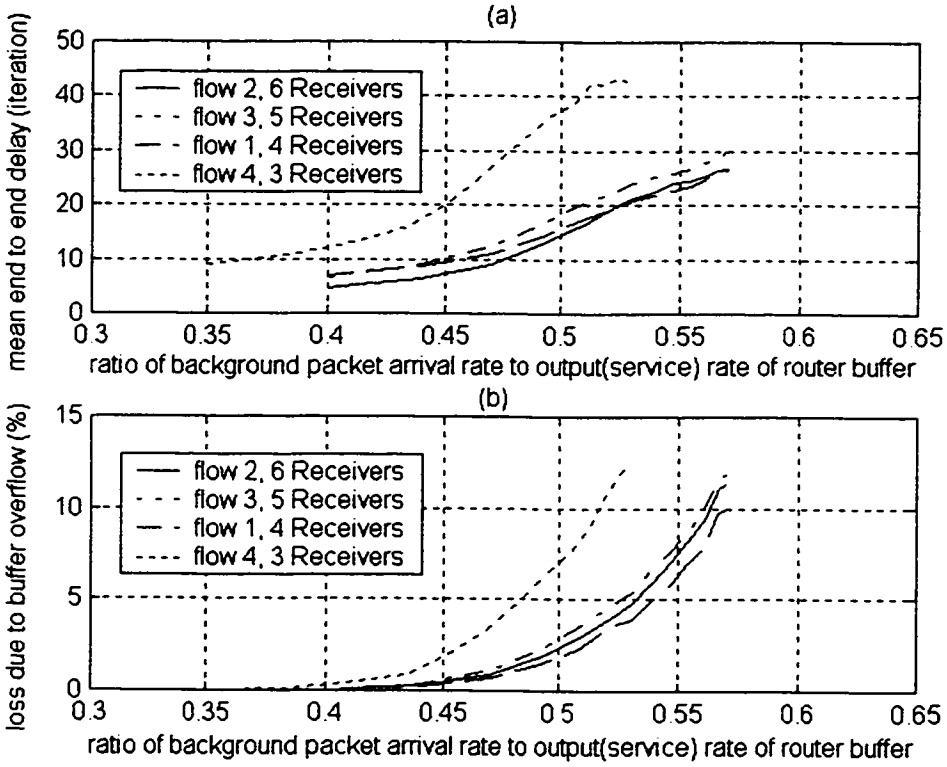


Figure 3.19: Hypothetical network. FEC/ARQ scheme. a. mean end to end packet delay and b. packet loss due to buffer overflow, versus ratio of background packet arrival rate to output rate of router buffer, while main flow and other multicast flow rates are constant and each equal to 0.25 output rate of buffer.

Figure 3.20 is the plot of no of transmissions versus packet loss due to buffer overflow. As it is seen FEC/ARQ scheme has less bandwidth usage compared to only ARQ protocol specially at higher loss rates and as the no of receivers per flow increases that is a proof of scalability of integrated FEC.

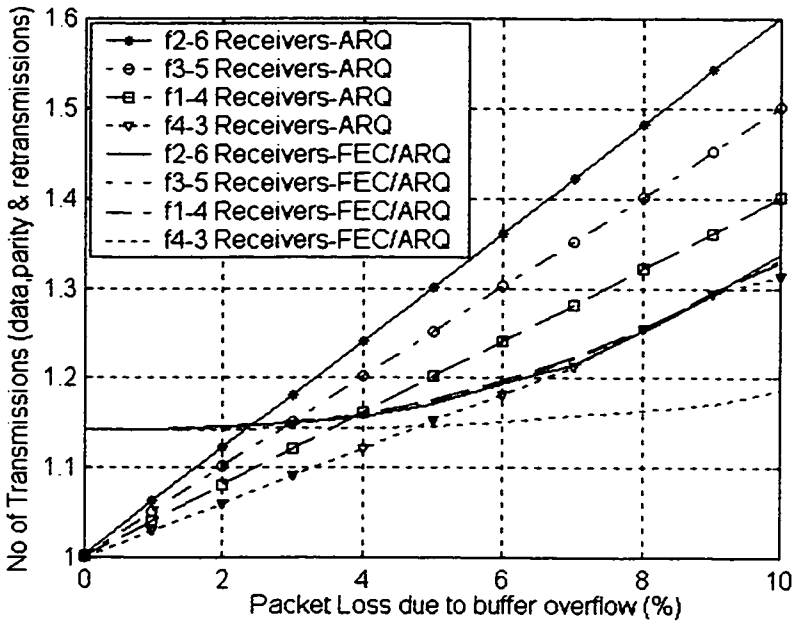


Figure 3.20: Hypothetical network. Comparison of FEC/ARQ and ARQ schemes. No of transmissions versus Packet loss due to buffer overflow.

Figure 3.21 is a plot of peak delay packet loss for two cases, one considering buffer delay and the other one without considering it. It is seen that peak delay for the case considering buffer delay is about 20 to 40 units higher. Peak delay unit in the figure is iterations. As mentioned previously, for a flow rate of 2Mbps each iteration is equal to 1 ms and maximum tolerable packet delay is 520 ms. This means that if a packet delay is more than 520 ms it is considered as lost since it is not received at receiver before relevant timer expires. In Figure 3.21delays are less than 520 i.e. no packet is considered lost due to excessive delay. In practice such long delays (20 to 40 ms) are not expected to happen in buffers across an end-to-end link and transmission rates of up to 2Mbps seem

achievable. Of course, if tolerable peak delays are exceeded the transmission rate has to be reduced.

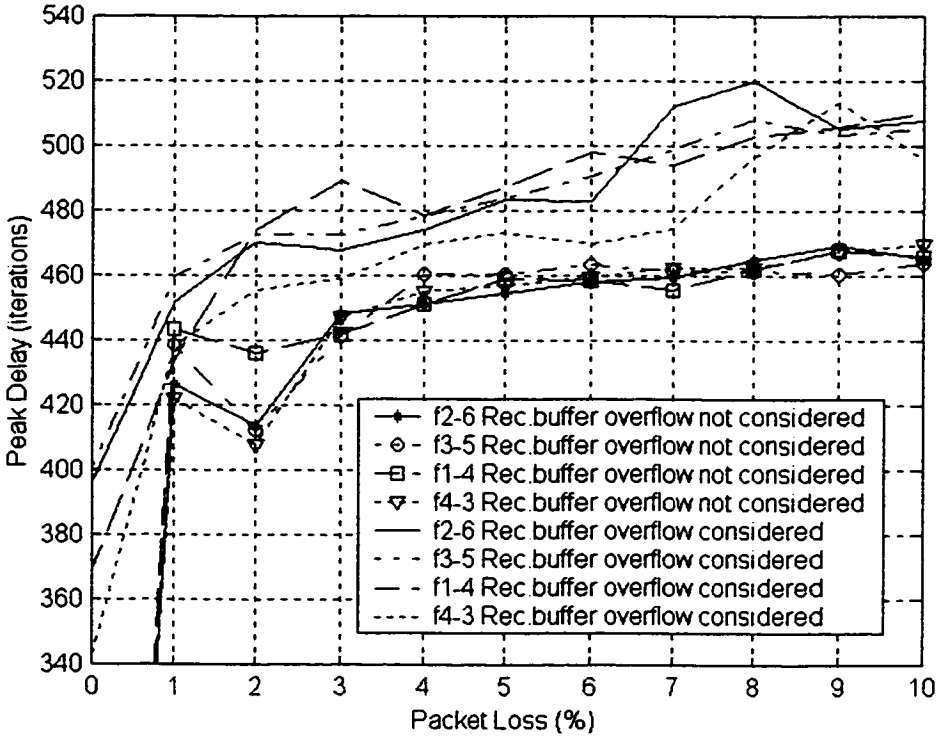


Figure 3.21: Hypothetical network. FEC/ARQ scheme. Peak delay versus packet loss.

Chapter 4

Conclusion and Future Work

4.1 Conclusion

The results achieved by laboratory testing our integrated FEC/ARQ code on two sender/receiver PCs show that the QOS technique addressed in this thesis performed reliably for random packet losses up to 10% and actual data flow rates up to 1.7 Mbps. This is achieved at extra cost of coding/decoding in the end systems. Coding delay in the sender can be reduced by 1) pre-encoding the packets off-line and storing the encoded information on disk prior to transmission; 2) using a more powerful machine in the sender; or 3) using dedicated hardware in the sender.

With our FEC/ARQ scheme, the receiver starts to play data after an initial start up delay of one window length (e.g. 500 ms for 2 Mbps transmission rate).

Error/erasure control feedback is very small for integrated FEC. NAK is sent for each block of information (consisting of N words) and only when packet loss rates are large, while for the schemes without FEC, NAK is sent for each missing packet. Number of retransmissions at worst case is less than one percent of total transmitted information, thus showing extremely minor usage of extra network bandwidth to achieve full reliability. In case of burst loss, performance of FEC is improved by interleaving encoded

words prior to transmission. Interleaving lets the sender spread the transmission of an encoded word over an interval that is longer than the burst loss length.

Simulations performed on our hypothetical multicast network results in the following. For independent packet losses at homogeneous receivers, by increasing the number of receivers, total no of transmissions (including retransmissions) for FEC/ARQ scheme is almost constant for a considerable range of packet losses and there is only a minor increase due to retransmissions at high losses (around 10%), while without FEC number of transmissions increases considerably. Thus it is concluded that while ARQ schemes suffer from weakness in scalability, FEC/ARQ protocol is scalable and can be used with large number of receivers. Also it is seen that for various flows in the network FEC/ARQ is completely reliable for losses below 5% and almost reliable up to 10% packet loss. Residual loss for ARQ with the same number of allowed NAKs/lost packet is at least 10 times the value for FEC/ARQ. Also for integrated FEC, number of NAKs is considerably less than the case without FEC.

For shared losses, the number of receivers needs to be larger before the benefits of integrated FEC appears. Simulations show that the overhead of transmitted parities with integrated FEC is amortized by the repair efficiency for greater number of receivers for shared losses than for independent losses. Other observations for independent losses continue to hold for shared losses and integrated FEC outperforms the case with only ARQ even when NAK avoidance is considered. The results for flows with heterogeneous receivers show that the performance of integrated FEC is determined by high loss receivers.

In the last section (3.4.4), the hypothetical multicast network is considered while losses are due to overflow of buffers and total packet delay also includes queuing delays in routers. The important effect is that packet delay has increased due to queuing delay in the buffers. In case, the packet delays are less than maximum tolerable delay, transmission rates of up to 2 Mbps (1.7 Mbps actual data rate) seem achievable. If in practice, due to delay in buffers, tolerable packet delays are exceeded the transmission rate has to be reduced.

In all the above cases our integrated FEC/ARQ scheme is shown to be scalable, reliable and utilizes network bandwidth efficiently and it outperforms schemes that do not use FEC. These advantages are achieved at cost of coding/decoding in the end systems.

4.2 Suggestions for Future Work

Expanding the proposed integrated FEC/ARQ code to cover a real multicast network consisting of a sender and a few routers and several receivers and practical testing and evaluation of the performance is the next step that is already being done at ETS network laboratory by other researchers.

In order to decrease the decoding delay and consequently the processing delay at the receiver, a faster and more efficient code is to be prepared.

Bibliography

[1] Sanjoy, Paul, "Multicasting on the Internet and Its Applications," Kluwer Academic Publishers, 2000.

[2] Larry L. Peterson & Bruce S. Davie, "Computer Networks A Systems Approach," Morgan Kaufmann Publishers. Inc., 1999.

[3] Kosiur, Dave, "IP Multicasting, The Complete Guide to Interactive Corporate Networks," John Wiley & Sons, Inc., 1998.

[4] Vicki Johnson and Marjory Johnson, "How IP Multicast Works," IP Multicast Initiative, www.ipmulticast.com

[5] http://www.ietf.org/rfc/rfc1112.txt?number=1112

[6] "Introduction to IP Multicast Routing", An IP Multicast Initiative White Paper, http://www.ipmulticast.com/community/whitepapers/introrouting.html

[7] M. Capan, "Survey of Different Multicast Routing Protocols," Proceedings of International Conference in multimedia and hypermedia systems, MIPRO'98, Opatija, Croatia, June 1998.

[8] Brian Neil Levin and J. J. Garcia-Luna-Aceves, "A Comparison of Reliable Multicast Protocols," 1998.

[9] B.J. Dempsey, J.C. Fenton and A.C. Weaver, "The Multidriver: A reliable Multicast Service Using Xpress Transport Protocol," Proceedings of 15th Conference on Local Area Networks, Minneapolis, Minnesota, Pages 351-358, September 1990.

[10] R. Talpade and M.H. Ammar, "Single Connection Emulation (SCE): an architecture for providing a reliable multicast transport service," Proceedings of international conference on Distributed Computer Systems(ICDCS)'95, October 95.

[11] W. Dang, "Reliable File Transfer in the Multicast Domain," Technical Report University of Hawaii, 1993.

[12] K. Miller, K. Robertson, A. Tweedly and M. White, "Star-Burst Multicast File Transfer Protocol (MFTP) Specifiction," Internet Fraft, draft-miller-mftp-spec 02.txt, January 1997.

[13] T. Shiroshita, T. Sano, O. Takahashi and N. Yamanouchi, "Reliable Multicast Transport Protocol Version2,"Internet Draft, draft-shiroshita-rmtpv2-00.txt, September1997.

[14] J. Macker and W. Dang, "The Multicast Dissemination Protocol (MDP) Framework," Internet Draft, draft-macker-mdp-framework-02.txt, May 1997.

[15] R. Yavatkar, J. Griffioen, and M. Sudan, "A reliable Dissemination Protocol for Interactive Collaborative Applications," Proceedings of ACM Multimedia'95, Pages 333-344.

[16] H. W. Holbrook, S. K. Singhal and D. R. Cheriton, "Log Based Receiver-Reliable Multicast for Distributed Interactive Simulation," Proceedings of ASIGCOMM'95, Pages 328-341, October 1995.

[17] M. Hofmann, "Enabling Group Communication in Global Network," Proceeding of global Networking'97, Calgary, Canada, June 1997.

[18] B. N. Levin and J. J. Garcia-Luna-Aceves, "A Comparison of Known Classes of Reliable Multicast Transport Protocols," Proceedings of International Conference on Network Protocols, October 1996.

[19] B. Whetten, T. Montgomery and S. Kaplan, "A High Performance Totally Ordered Multicast Protocol," Theory and Practice in Distributed Systems K. P. Birman, F.Mattern, A. Schiper (Eds) Springer Verlag LNCS938.

[20] S. Floyd, V. Jacobson, S. McCanne, C-G. and L. Zhang, "A reliable Multicast Framework for Light-weight Sessions and Application Level Framing," Proceedings of ACM SIGCOMM'95, Pages 342-356, October 1995.

[21] Jose F. de Rezende and Serge Fdida, "Scalability Issues for Reliable Multicast Protocols," Technical Report, Piere and Marie Curie University, Paris, France, 1999.

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

[23] Luigi Rizzo, "Effective Erasure Codes for Reliable Computer Communication Protocols," ACM SIGCOMM Computer Communication Review, Vol. 27, No. 2, April 1997.

[24] Salvatore Gravano, "Introduction to Error Control Codes," Oxford University Press, 2001.

[25] C. Huitema, "The case for packet level FEC." in Proc. IFIP 5th Int. Workshop on Protocols for High Speed Networks (PfHSN'96), Sophia Antipolis, France. Oct. 1996, pp. 110-120.

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

[27] J. Metzner, "An improved broadcast retransmission protocol," IEEE Trans. Commun., vol. COM-32. pp. 679683, June 1984.

[28] Don Towsley, Jim Kurose and Sridhar Pingali, "A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols," IEEE Journal on Selected Areas in Communications, Vol. 15, No. 3, April 1997, pp. 398-406.

[29] Phil Karn's Home Page, http://www.ka9q.net/code/fec/

[30] Stephen B. Wicker, "Error Control Systems for Digital Communication and Storage," Prentice Hall Inc., 1995.

[31] Jyh-Hong Jeng and Trieu-Kien Truong, "On Decoding of Both Errors and Erasures of a Reed-Solomon Code Using an Inverse-Free Berlekamp-Massey Algorithm," IEEE Transactions on Communications, 47(10): 1488-1494, Oct. 1999.

[32] C.Kenneth Miller, "Reliable Multicast protocols and applications," Internet Protocol Journal, Cisco Publications, September 1998.

[33] Ahmed Elhakeem, Tarek N. Saadawi and Mostafa H. Ammar, "Fundamentals of Telecommunication Networks, John Wiley & sons, inc., 1994.

[34] Richard D. Gitlin, Jeremiah F. Hayes and Stephen B. Weinstein, "Data Communication Principles," Plenum Press, 1992.

[35] Pedro Paulo Ferreira Bueno and Antonio Pires de Castro Junior, "Introduction to socket Programming," Published on line in Issue 47 of Linux gazette, Nov. 1999, http://www.linuxgazette.com/issue47/bueno.html

[36] Michael J. Donahoo and Kenneth L. Calvert, "The Pocket Guide to TCP/IP Sockets," Academic Press, 2001.

[37] William A. Shay, "Understanding Data Communications and Networks, Brooks/Cole Pub. Co., 1998.

[38] L. Hughes and M. Thomson, "Implosion-Avoidance Protocols for Reliable Group Communications," in Proc. Of 19th Conference on Local Computer Networks, (Mineapolis, USA), Oct. 1994.

[39] Jorg Nonnenmacher and Ernst W. Biersack, "The Impact of Routing on Multicast Error Recovery", Computer Communications, Vol. 21, No. 10, pp. 867-879, July 1998.