Layered Hybrid ARQ/FEC for Increasing Quality of Service for MPEG 1-2 Video Communication over Networks: Design and Practical Implementation

Joud Abdel Messie

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfilment of the Requirements
For the Degree of Master of Applied Science at
Concordia University
Montreal, Quebec, Canada

February 2003

0-612-77682-4

Abstract

Layered Hybrid ARQ/FEC for Increasing Quality of Service for MPEG 1-2 Video Communication over Networks: Design and Practical Implementation

Joud Abdel Messie

This thesis details the design and implementation of a new technique, called the Layered Hybrid ARQ/FEC, for increasing the Quality of Service (QoS) of MPEG video bitstreams while they are transmitted over networks and enduring data loss. This new technique, which is the major focus of this thesis, can be achieved by first combining the two error control methods: Automatic Repeat Request (ARQ) with RS-FEC (the Reed Solomon Forward Error Correction method), and second by defining different layers of protection for each type of frame in the MPEG video stream. This thesis starts by explaining the structure of MPEG video bitstreams and the different levels of importance of each type of frame (I, P, or B frames). Also, the thesis explains in details some ARQ and FEC techniques, focusing on Reed-Solomon mechanism. The thesis then introduces the Layered Hybrid ARQ/FEC technique being applied on MPEG video; the method is first theoretically detailed and then implemented as a server/client application-layer software using C language. Data are collected, and analysed, and compared with other methods. Finally, conclusions are derived and suggestions for future improvements are highlighted.

Acknowledgements

I would like to present the deepest and sincerest thanks to Dr. Ahmed El Hakeem, for without his guidance, encouragement, knowledge, and patience this thesis would not have seen the light.
Many thanks to all the people that helped and advised me to make this thesis as perfect as possible: Dong Chen, Naser Shayan, and Jee lee.
Special thanks to Dr. William Lynch and the group of ETS who provided their equipment for testing and experimenting, especially Dr. Michel Kadoch and Maria Bennani.

Dedication

To my family, Naji, Majd, Rawad, and Mary; without love, forgiveness, and prayer no small step is possible on the road

To my Kristine; my present fiancée, my future wife, and my love forever

To Tarek; small pushes from true friends are the shortest way to success

To Marcel and Ziad; when the heart is filled with joy, imagination is unlimited

Table of Contents

List of Figures

List of Tables

Abbreviations

ACK	Acknowledgement
ARQ	Automatic Repeat Request
ATM	Asynchronous Transfer Mode
B Frame	Bi-directional Frame
BCH	Bose-Chaudhuri-Hocquenghem
BER	Bit Error Rate
CCIR	International Radio Consultive Committee
CRC	Cyclic Redundancy Check
CUDP	Complete UDP
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DL	Data Length
FEC	Forward Error Correction
FTP	File Transfer Protocol
GBN	Go-Back-N
GF	Galois Fields
GOP	Group of Pictures
HDTV	High Definition Digital TV
I Frame	Intra Frame
IDCT	Inverse Discrete Cosine Transform
IEC	International Engineering Consortium
IP	Internet Protocol
ISO	International Organisation of Standardisation
LAN	Local Area Network
MB	Macro Block
MC	Motion Compensation
MLD	Mean Luminance Difference
MPEG	Moving Pictures Expert Group
MSE	Mean Square Error

NAK	Negative Acknowledgement
OSI	Open System Interconnection
PET	Priority Encoding Transmission
P Frame	Predictive Frame
PS	Program Streams
PSNR	Peak Signal to Noise Ratio
QoS	Quality of Service
RS	Reed-Solomon
SMTP	Simple Mail Transmission Protocol
SNR	Signal to Noise Ratio
SR	Selective Repeat
SW	Stop and Wait
TCP	Transmission Control Protocol
TPDU	Transport Protocol Data Unit
TS	Transport Stream
UDP	User Datagram Protocol
VBR	Variable Bit Rate
VLC	Variable Length Code
WAN	Wide Area Network

CHAPTER 1

Introduction

1.1 Background

The Internet is based on the principle of the connectionless Internet Protocol IP. All datagrams that are meant to be sent from one IP host/server to another are routed on a hop-per-hop basis over the backbone routers. As with the protocols that have the OSI (Open System Interconnection) architecture model, this network layer IP does not make any assumptions about the underlying layer and does not give any guarantee of Quality of Service (QoS). In other words, in situations of congestion and overwhelming traffic some packets are liable to be dropped or ignored. While some applications could tolerate a minor loss in datagrams, others could not. Thus, end-to-end quality assurance becomes the responsibility of higher level protocols (such as the transport layer TCP: Transmission Control Protocol) which uses positive acknowledgements, 3-way handshaking, retransmission requests ARQ, CRC checking, and traffic flow control methods to ensure that correct reception are made for all packets. However, this guarantee has its disadvantages too; it imposes time delays and jitter and consumes more bandwidth and network resources. The mechanisms of positive acknowledgements and Automatic Repeat reQuests (ARQ) and the delay they impose are easy to tolerate if the application is insensitive to time delays (such as the Electronic Mail Protocol SMTP and File Transport Protocol FTP).

A simple ACK/ARQ mechanism can be explained in short as follows. As soon as the end-user receives a packet, it acknowledges the sender with a datagram called ACK meaning that this specific packet was received error free [10]. On the other hand, when the receiver senses data loss, it locates the missing datagrams and sends a Negative Acknowledgement (NAK) for the server demanding a retransmission. The server checks whether these requested packets are still available for retransmission and if so, it prepares all the data and send them for the second time. In some protocols, receivers keep sending NAK as long as data is missing.

It is not hard to notice that this ARQ method would lose its credibility and become inefficient, if not unacceptable, when huge amounts of retransmission are requested by different users, because it may lead to data implosion in the network and to more delays. This is especially the case in *multicast* sessions, in which different network paths (channels) have different and uncorrelated dropping and error patterns.

With the emergence of interactive real time multimedia applications (such as video conferencing and video-on-demand), tighter time constraints with higher Quality of Service were imposed and demanded. To achieve what- seems -to -be contradictory goals, keeping in mind the network's bandwidth limitations, engineers started to think of alternatives to simple ARQ methods. *Forward Error Detection and Correction* algorithms were introduced as channel encoding stage, such as Reed-Solomon and BCH algorithms. Some of these schemes will be discussed in more details in section 3, but for now the basic idea behind these algorithms is shown.

Forward Error Correction (FEC) involves the addition of some redundant blocks of data (bits, bytes, or even packets) that can be used to aid in correcting any error in the received datagrams. Shannon theorem states that there always exists a coding scheme that enables information to be transmitted over any given channel with arbitrarily small error probabilities provided that the data rate is less than the channel capacity [7]. This means that if some bytes are corrupted, because of whatever reason, the receiver could recover the correct data without asking for retransmission from the sender. As a result this would decrease the probability of having NAK/ARQ implosion in the network. The drawback of this method is that it creates an overhead in data transmission, and it would waste the networks' resources if the error rate of the channel were very small.

A better solution would be to integrate the two previous methods to benefit from their advantages without overloading the network or depleting its resources. These methods are called *Hybrid FEC/ARQ* method and they are the major focus of this thesis. This thesis applies Hybrid FEC/ARQ over MPEG-1/2 compressed video streams methods to achieve the best QoS at the receiving end.

MPEG, which is the abbreviation of Moving Pictures Expert Group - an International Organisation of Standardisation (ISO) group, has developed a series of audio-visual standards known as MPEG-1, MPEG-2 and recently MPEG-4. [3]. They are, in principle, standards of compressing video images and audio sounds that allow internetworking between video communication equipment from different vendors. MPEG-1 is directed to the audio/video CD-ROM applications, while MPEG-2 (which is a superset of MPEG-1) forms the basic element of existing and future digital TV and adopted for High Definition Digital TV (HDTV). On the other hand, MPEG-4 is a low-bit-rate compression standard that is to be used for future wired and wireless multimedia communication. The thesis will focus on MPEG-1/2 since they attracted much attention over the years and they are the most common forms of compressed video that exist in the Internet world.

The thesis will include an overview of the MPEG 1/2 standards, focusing on the video part of them, ignoring audio. Then a review of various Error Control Codes while focusing on Reed-Solomon codes and ARQ will follow. Finally, the thesis will include a practical programming implementation, using C language, of a client/server relation in a Hybrid FEC/ARQ -using Reed -Solomon codes- environment. The major issue is to show the effects of applying different levels of FEC/ARQ protection based on the importance of each MPEG2 frame in a video. As will be shown later in this thesis, MPEG video frames are not of the same importance for QoS. I frame, for example, is the most important frame because it plays the role of the reference for encoding the other two picture types, B and P. This means that a small error in an I Frame will not only affect this frame, but will also affect all the other frames that are referenced by it. Consequently the error propagates and hits many frames and QoS degrades. On the other hand, losses in B Frames would not propagate since they do not reference any kind of frames. Consequently, giving high levels of FEC protection to I Frames would eventually increase the QoS of the video in lossy channels. This layered FEC protection of each kind of frame is discussed thoroughly in this thesis.

1.2 Thesis Organisation

In Chapter 2, an overview of MPEG2 standard is introduced. The chapter will discuss methods of compression, Digital Cosine Transform (DCT) for compression in space and motion compensation for compression in time. Furthermore, the different types of pictures are fully explained with the properties of each one and the effects of errors on each type. The different effects of errors on different types of frames are the basic brick of the layered Hybrid ARQ/FEC encoding method, which is aim of this thesis. Finally, system and video layers are described, along with the levels, profiles, and start codes of MPEG-2 video streams.

Chapter 3 will include a short review of the basic idea from Error Control methods. ARQ methods and its different types are explained as well as many FEC algorithms, including convolutional and block codes. The main attention would be directed to Reed-Solomon codes, which would be implemented via the C program in Chapter 4. Finally, the basic techniques of combining ARQ and FEC are mentioned.

Chapter 4 will include a practical implementation of Layered Hybrid ARQ/FEC, applied on MPEG video streams. Reed-Solomon is the FEC method used along with ARQ in a C program, and real time test is performed over a client/server Linux systems. Data and collected and results were got.

In Chapter 5, the performance of this approach is evaluated. The results that were taken in Chapter 4 will be compared with other algorithms such as non-layered Hybrid ARQ/FEC. Moreover, some enhancements are added to the new approach to decrease the number of packets needed to be retransmitted. At the end of the chapter, comparisons between this approach and other approaches implemented previously would be done

Finally, in Chapter 6, the limitations of this approach would be stated and some conclusions and future work would be discussed.

CHAPTER 2

MPEG Standard Overview

2.1 Introduction

In the early stages of the communication age, almost all the electronic communication systems started as analog systems; the sender sends voltage variations which the receiver uses to control transducers such as loud speakers, and TV cathodes. With the introduction of the 0 and 1 *bits* as a way of representing any type of data, including signals, the *digital revolution* started. Digital information can be stored and recovered, transmitted and received, processed and manipulated virtually without error. Cheaper and more robust digital systems were introduced and they are soon to replace analog systems. Digitised entertainment was born and audio disks (CDs) almost replaced cassette tapes, and digital video satellite will soon make analog satellite history. Digital video and audio signals integrated fast with computers and added another dimension to the digital entertainment era. This digital multimedia fever will hit the also all the new communication systems as Internet and LANs (Local Area Networks), WANs (Wide Area Networks), wireless cell phones, cable television, etc....

However, the growth of communication systems and number of users made faster transmission and reception necessary, especially for real-time multimedia application that can not afford time delay, such as video conferencing. *Compression* methods were introduced and applied on audio and video. These methods attempt to benefit from the amount of *statistical redundancy* that exists in audio and video bitstreams. That is, consecutive samples are strongly correlated so that one sample can be predicted fairly accurately from another. The biggest advantage of compression is in data rate reduction. Data rate reduction reduces transmission costs and where a fixed transmission capacity is available, results in a better quality of multimedia presentation [2]. Moreover, compression reduced the amount of needed hardware - storage area for multimedia application. MPEG standards are one method of compressing audio and video bitstreams.

MPEG (Moving Picture Expert Group) was started in 1988 as a working group within ISO/IEC with the aim of defining standards for digital compression of audio-visual signals. MPEG's first project, MPEG-1, was published in 1993 as ISO/IEC 11172. It supports video coding up to 1.5 Mbps giving quality similar to VHS and stereo audio at 192 Kbps. It is used in the CD-I and Video CD systems for storing video and audio on CD-ROM. However, during the 1990's MPEG group realised the need for a second standard for broadcast application with a higher data rate. This was called MPEG-2. It is capable of coding standard definition television at bit rates from about 3-15 Mbps and high-definition television at 15-30 Mbps [1]. MPEG-2 is merely an extension of MPEG 1 features and it is the reason why it is considered a superset of MPEG-1. The most recent standard given by the MPEG group is MPEG-4, which was initially targeting video conferencing applications with low bit rates. However its role was expanded and it became efficient in sending data in rates ranging from few Kbps to few Mbps [31].

Throughout this paper, the major focus will be MPEG-2, and hence MPEG-1 since it is a subset of it. MPEG-4, however, will not be discussed in this paper since it differs in a major way from the previous two standards, and since MPEG1-2 are the most common compression methods used for Internet and network applications. It is also worth noting that the audio part of the standard would be of a little importance and the major interest would be the video part.

2.2 MPEG-2 Standard Specifications

MPEG-2 standard is formally referred to as ISO 13818 and it consists of the following parts: [2]

- 113818-1: Systems
- 113818-2: Video
- 113818-3: Audio
- 113818-4: Conformance
- 113818-5: Software
- 113818-6: Digital Storage Media -Command and Control (DSM-CC)
- 113818-7: Non Backward Compatible Audio
- 113818-8: 10-Bit Video (this item has been dropped)
- 113818-9: Real Time Interface
- 113818-6: Digital Storage Media -Command and Control (DSM-CC) Conformance

Only the video and systems part will be explained in this overview

2.2.1 MPEG-2 Video Compression

It should be noted that MPEG standards do not describe the encoding process; instead, the standard specifies the data input format for the decoder as well as detailed specifications for interpreting the data [4].

The size of a standard digital-television, recommended by CCIR[1] 601 is 720 X 576 pixels at a frame rate of 25 MHZ. MPEG-2 aimed to match this resolution, with a quality of picture no less than that of NTSC/PAL. However, MPEG has many profiles and levels that support flexible and different size of resolutions, bit rates and frequencies. Profiles and levels of MPEG-2 are discussed later in this chapter

An MPEG -2 encoder first samples the video image at a rate of 30 frame per second to get a sampled-image, which is formed of sequence of *pixels*. Red, Green and Blue (RGB) signals coming from a digital camera can be represented as Luminance (Brightness) (Y) and Chrominance (UV) components. CCIR-601 recommends how the component YUV video signals can be digitised to form the pixels of the frames of the sampled image. Since the human eye is more sensitive to brightness changes than to chromaticity changes, CCIR-601 recommends 4:2:2 and 4:2:0 terms, which represent the sample structure of YUV, respectively, in a digital picture. 4:2:2 indicates that the chrominance is horizontally sampled by a factor of two, relative to luminance; 4:2:0 indicates that the chrominance is sampled horizontally and vertically by factor of two relative to the luminance [1]. Computing the bit rate necessary to achieve a digital television quality, taking into consideration the two YUV:

$4:2:2: 720 \times 576 \times 25 \times 8 + 360 \times 576 \times 25 \times (8+8) = 166 Mbps$

$4:2:0: 720 \times 576 \times 25 \times 8 + 360 \times 288 \times 25 \times (8+8) = 124 Mbps$ [1].

That is when every pixel is represented by 8 bits. It can be seen directly that this rate is too high to be used on some communication systems such as Internet or wireless LANs. Here comes the role of MPEG-2 compression algorithm. To achieve the best a

[1] CCIR is the International Radio Consultive Committee

reduction of bit rate, the encoder removes the redundant information in the signal prior transmission. Pixel values in images (whether video or still images) are *not* independent; they are strongly correlated with their neighbours within the same frames, that is called *spatial redundancy*, and across frames, and that is called *temporal redundancy* [1]. To eliminate spatial redundancy, intraframe *Discrete Cosine Transform (DCT)* coding is used and to remove temporal redundancy, interframe motion compensation algorithm is applied.

2.2.1.1 DCT for Eliminating Spatial Redundancy

DCT is very closely related to Discrete Fourier Transform DFT to the extent that the coefficients of DCT can be given frequency interpretation [3]. The two-dimensional DCT for an N by N image block would give an N by N block of coefficients. The reduction occurs in the number of bits follows from the observation that, for typical blocks of natural images, the distribution of coefficients is not uniform; meaning that the higher coefficients are concentrated near the DC value and low frequencies. DCT tends to concentrate the energy around lower frequencies [1]. A common practice is to apply DCT over non-overlapping 8x8 pixel blocks to get 8x8 DCT. A typical image DCT can be shown in Figure.1

The *NxN* DCT is defined by:

$$F(u,v) = \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\frac{(2x+1)u\pi}{2N} \cos\frac{(2y+1)v\pi}{2N}$$

$$C(u),C(v) = \begin{cases} \frac{1}{\sqrt{2}}\, for & u,v=0 \\ 1 & otherwise \end{cases}$$

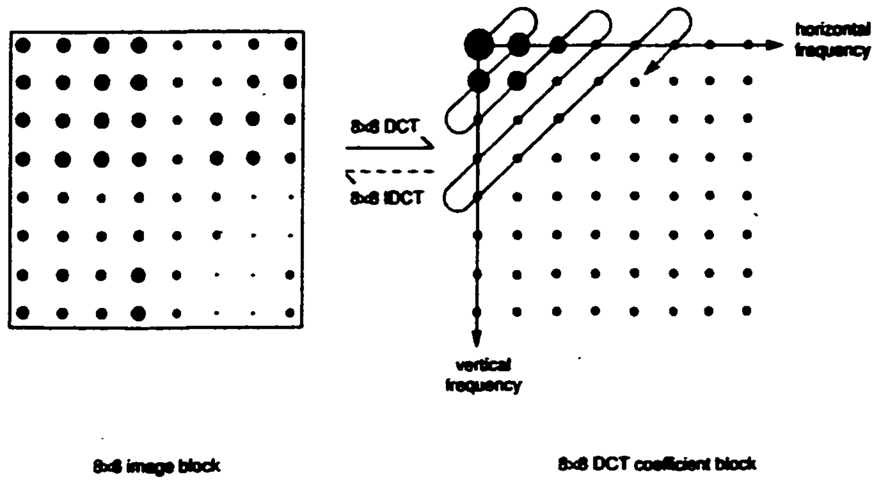


Figure 1: Discrete Cosine Transform (DCT) over an 8x8 image block [1]

The inverse DCT (IDCT) is defined as:

$$f(u,v) = \frac{2}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} C(u)C(v)F(x,y)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N}$$

where x,y are the spatial co-ordinates in the image block, u,v are the co-ordinates in the DCT coefficient block and *C(u) and C(v)* are weighting functions.

After the DCT phase, the MPEG coder performs a *quantization* process in an efficient manner. Since the human eye is more sensitive to lower frequencies, the higher frequency coefficients will be quantized more coarsely than the lower ones. Then, the quantized block is scanned in a zig-zag manner (as Figure 1 shows) and then the values scanned are *statistically* encoded (entropy encoded) using *variable length codes* VLC. VLC allocates different codewords different bit lengths, according to their probability of occurrence [1]. Figure 2 shows a block diagram for a DCT based encoder (it includes motion estimation, which would be explained in the next section.

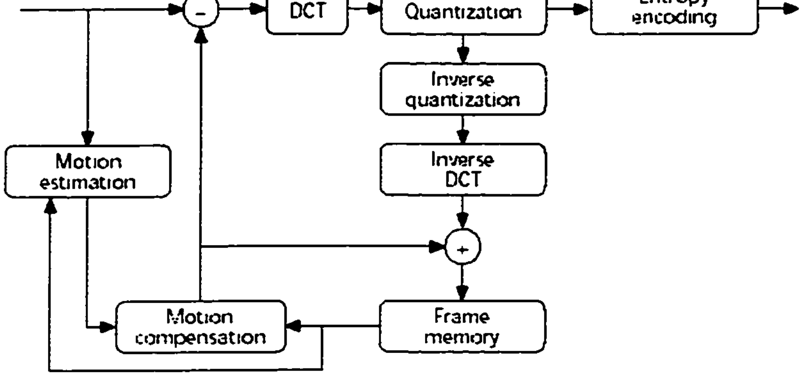


Figure 2. DCT /Motion Estimation encoder [5]

2.2.1.2 Motion Compensation for Eliminating Temporal Redundancy

Consecutive frames of video are highly correlated within time. If this temporal redundancy was eliminated, bit rates in MPEG2 video can noticeably decrease. The technique to eliminate this redundancy is to attempt to predict the frame to be encoded from a previous *reference* frame. The basic unit for temporal prediction in MPEG-2 is the *MacroBlock* (MB), which consists of 16x16 non-overlapping pixels. This prediction is easily done in stationary areas, but for moving areas *Motion Compensation* (MC) is applied. MC works by offsetting any translational motion that has occurred between the block being coded and the reference frame, and to use a shifted block from the reference frame as the prediction. The positions of the best matching prediction MacroBlocks are indicated by Motion Vectors that describe the displacement between them and the Target MBs. The motion vector information is encoded along with the prediction error (the difference between the prediction of the frame and the frame itself) and transmitted together [2].

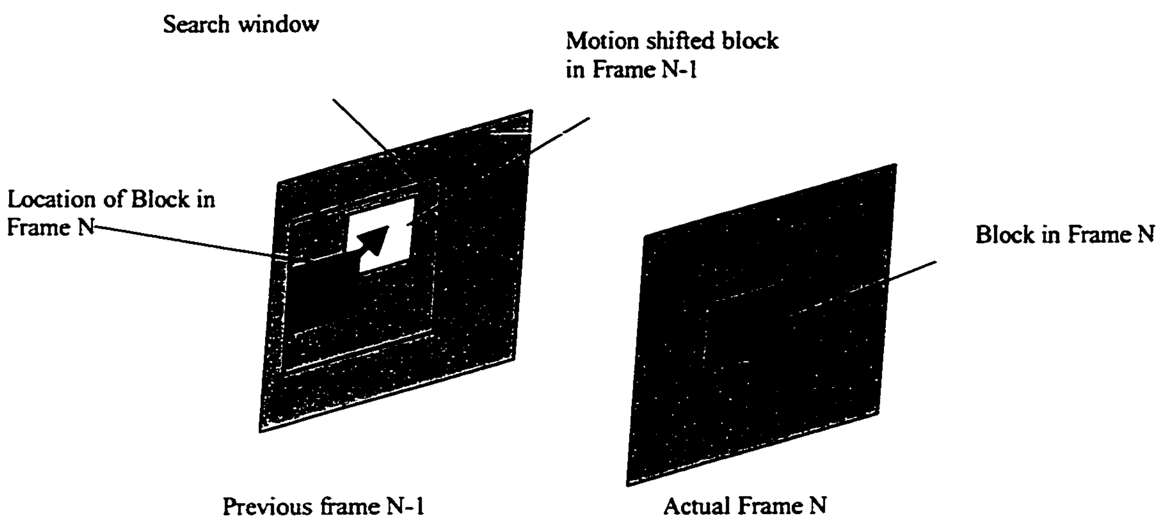

Search window
Motion shifted block in Frame N-1
Location of Block in Frame N
Block in Frame N
Previous frame N-1
Actual Frame N

Figure 3: Only one motion vector is estimated, and coded, and transmitted for each Macroblock [3]

2.2.1.3 Picture or Frame Types in MPEG-2

The previous two encoding methods that eliminate temporal and spatial redundancy are applied over digital video frames to get one of three modes or types of MPEG-2 frames [15]:

a) *Intra (I) Frames*: I Frames are coded without reference to other pictures; only spatial redundancy is eliminated and no motion compensation, for eliminating temporal redundancy, is applied. Moderate compression ratio is achieved. These I Frames are used as references for the other two types of pictures.

b) *Predictive (P) Frames*: P Frames are coded by applying *forward prediction*; they are motion- compensated with reference to *previous* I Frames and P Frames, and the residual difference is coded applying DCT, in much the same way I Frames are encoded. Higher compression rates are achieved in P frames.

c) *Bidirectional (B) Frames:* B Frames are coded by applying *forward and backward prediction*; they are motion-compensated with reference to *previous and future* I and P Frames. Highest compression rates are achieved in B frames. To enable backward prediction, the coder reorders the pictures from natural 'display' order to 'bitstream' order so that the B picture is transmitted *after* the previous and next pictures it references.

The three different types of frames usually occur in groups, which are called Group of Pictures (GOP). Each GOP consists usually of one I Frame and many P and B Frames, and every GOP is represented by 2 parameters: N which is the number of Frames in the GOP and M which the spacing of P Frames. Figure 4 shows the reordering process in one GOP.

2.2.1.4 MPEG-2 Bitstream Structure

Every MPEG-2 encoded video stream consists of six layers; they are from top to bottom [6]:

1) Sequence layer: consists of many GOPs
2) Group of Pictures layer: consists of a number of different frame types
3) Picture layer: consists of many slices horizontal grouping of Macroblocks
4) Slice layer: consists of many Macroblocks
5) Macroblock layer: consists of number of luminance and chrominance blocks
6) Blocks layer: consists of 8x8 pixels

Each layer in the video stream can be identified by a unique start header for each level. Moreover, each component in each level has its distinguishing identifier too.

The layers of a video stream are shown in Figure 5 b. and Figure 5.a shows the video structure of MPEG-2.

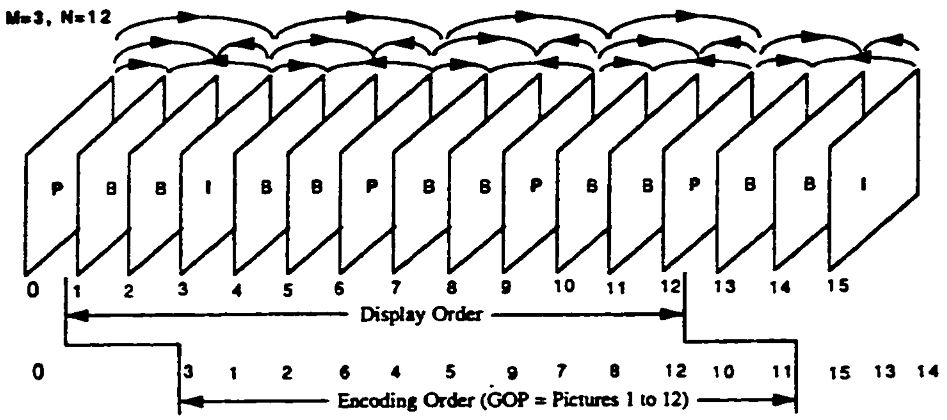


Figure 4:Example of a Group of Pictures [2]

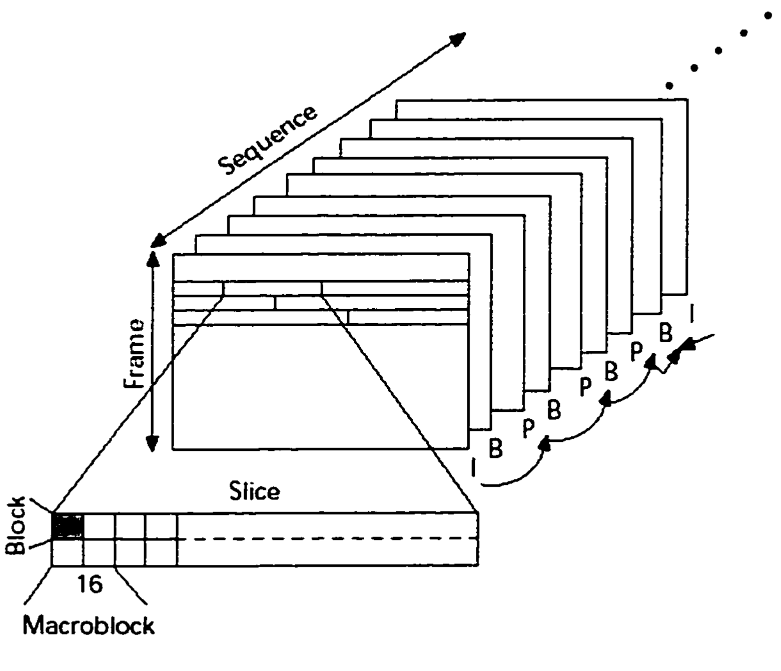


Figure 5a. . MPEG2 video structure [5]

2.2.2 MPEG-2 System Layer

An MPEG-2 compressed video bitstream cannot stand alone, since video alone is not considered valid information. The system layer usually resides on top of the video layer and plays a *wrapper* role for the video bitstreams. The major function of the system layer is to synchronise and multiplex the audio and video bitstreams into an integrated data systems. The MPEG-2 systems standard specifies two types of streams: *program streams and transport streams.*

Program streams (PS) use long variable length packets to multiplex and synchronise bitstreams with common time base. PS are used in storage applications such as DVDs. They are not ideal for transport since long packets are more susceptible to errors [4]. Transport streams multiplex streams that do not have common time base. Their packets have fixed length (188 bytes), making it efficient for transmission. It should be noted that the TS took into consideration Asynchronous Transfer Mode (ATM) data link layer. In the rest of this thesis, all the MPEG-2 streams are considered to be PS type. TS type is not to be dealt with since the ATM factor is not to be taken into consideration. Figure 6 shows the layers of the systems layer in MPEG.

2.2.3 MPEG -2 Profiles, Levels and Start Codes' Tables

MPEG-2 is a very big standard. Other from the tools of MPEG-1, it defines new tools and syntax for efficiently coding interlaced and scalable video. Considering this fact, full syntax and details of the standard may not be practical for all applications. MPEG-2 introduced the concept of *profiles and levels*. Profiles define subsets of algorithms and functionalities that fit differently many applications, each

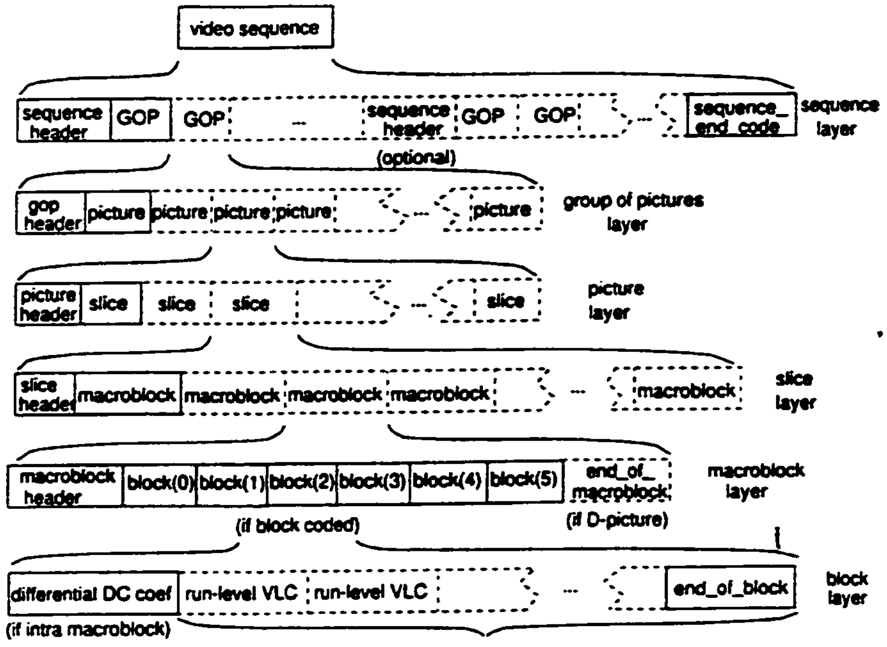


Figure 5b Layers of MPEG video stream [6]

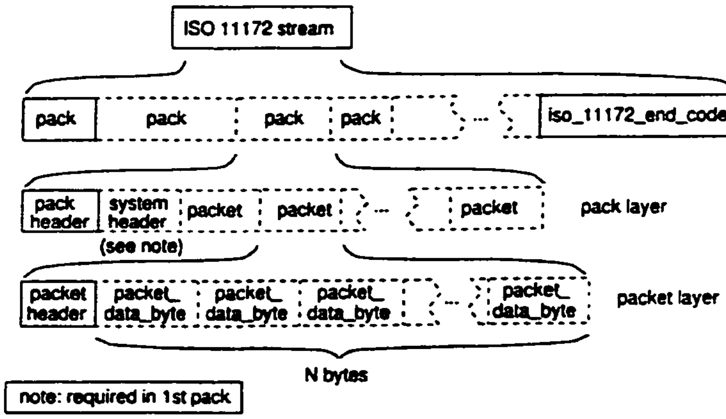


Figure 6. MPEG system layer [6]

according to its necessities. Each profile is a subset of the higher profiles. For example, Simple profile does not support B Frames prediction, while the higher Main profile does support B Frames. MPEG-2 sets of profiles are shown in Table 1.Levels are also sets that are used to put constraints on some of the parameters, like frequency of sampling and play rate and other values. Table 2 shows the levels of MPEG-2.

In Table 3, start codes for the video and system layers are displayed in numeric order. Those values of headers will play a key role in detecting the different kinds of frames - whether I, B or P- to enable layered channel-encoding protection levels. It should be clear that each sublayer in video and systems layers has its own start code that distinguishes it from other layers.

Profile	***Algorithm***
HIGH	Supports all functionality provided by spatial scalable profile plus: • 3 layers with the SNR and Spatial scalable coding mode • 4:2:2 YUV representation
Spatial Scalable	Supports all functionality provided by SNR scalable plus: • Spatial scalable coding (2 layer allowed): • 4:0:0 YUV representation
SNR Scalable	Supports all functionality provided by the Main profile plus: • SNR scalable coding • 4:2:0 YUV representation
Main	Non-scalable coding which supports: • Coding of interlaced video • Random access • B-Picture prediction • 4:2:0 YUV representation
Simple	Includes all functionality provided by main BUT: • Does not support B-Picture prediction • 4:2:0 YUV representation

Table 1. MPEG-2 profiles [3]

2.3 Propagation of Errors in MPEG -2

As MPEG-2 bitstreams are transmitted through any channel, wired or wireless, they are vulnerable to errors and losses. Any error or loss of any packet of the video bitstream will be reflected on the picture played. Since MPEG-2 video compression methods depend on eliminating temporal and spatial redundancy, some losses in

Level	***Parameters***
HIGH	1920 samples/line 1152 lines/frame 60 frames/s 80 M bits/s
HGH 1440	1440 samples/line 1152 lines/frame 60 frames/s 60 M bits/s
Main	720 samples/line 576 lines/frame 30 frames/s 15 M bits/s
Low	352 samples/line 288 lines/frame 30 frames/s 4 M bits/s

Table 2. Levels of MPEG-2 standard [3]

Start Code name	***Hexadecimal***	***Binary***
Video start codes		
Picture_start_code	00000100	00000000 00000000 00000001 00000000
Slice_start_code1	00000101	00000000 00000000 00000001 00000001
...	...	...
Slice_start-code175	000001AF	00000000 00000000 00000001 10101111
Sequence_header_code	000001B3	00000000 00000000 00000001 10110011
Group_start_code	000001B8	00000000 00000000 00000001 10111000
System start codes		
Packet-start_code	000001BA	00000000 00000000 00000001 10111010
System_header_start-code	000001BB	00000000 00000000 00000001 10111011

Table 3. Some MPEG-2 start codes [6]

certain frame types could propagate and hit another areas in the same frame, or even following frames. For example, if some packets of an I Frame are dropped due to congestion, errors will spread within the same picture up to the next synchronisation point (e.g. picture or slice header) due to DCT and variable length coding. This is called *spatial loss propagation*. Moreover, since the following B and P pictures in the GOP are coded with reference to this I Frame, the predicted values of some blocks in the receiver would deviate from the correct values and errors occur. This is called *temporal loss propagation*. Figure 7 shows data loss propagation in MPEG-2 videos.

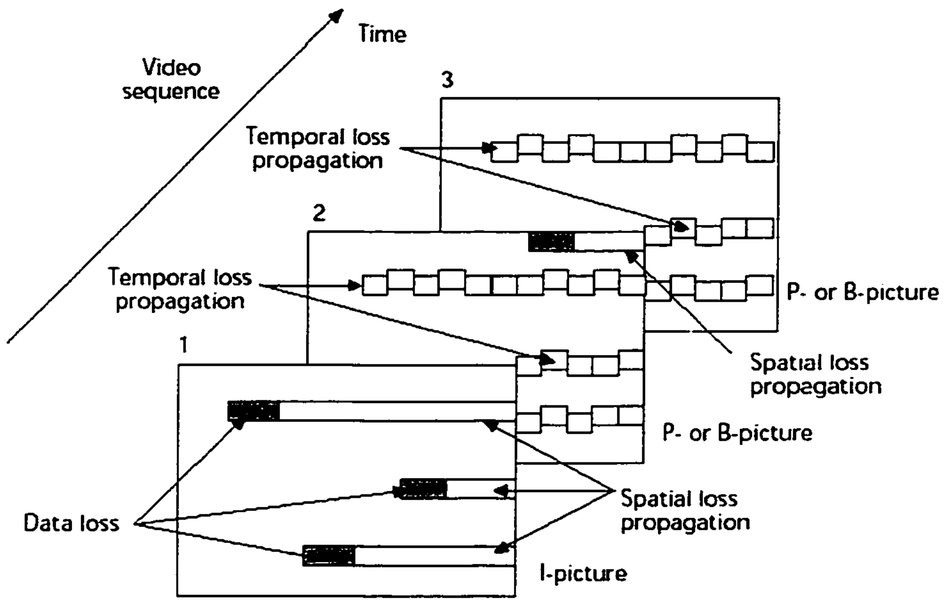


Figure 7: Data loss propagation in MPEG-2 [5]

CHAPTER 3

Error Control Methods

3.1 Introduction

The interest in error control methods and algorithms is increasing rapidly while designing and implementing communication systems. Losses and errors are sure to hit any stream of data while it is transmitted and received; that is the only fact that is certain in any communication system. These losses and errors would directly be reflected on the quality of service. For the receiver to recover from losses or dropped packets, it should apply at least one *error detection or correction method.* Choosing the appropriate recovery scheme depends basically on the nature of the application and its various constraints. For example, electronic mailing application (e.g. using SMTP) is concerned basically with delivering the complete message error-free, regardless of how much time this process would take; i.e. time delay and jitter are not the controlling criteria for QoS. On the other hand, in *IP telephony* systems, quality of voice, time delays and jitters are of equal importance.

Error control algorithms were introduced, each taking into consideration one or more controlling factors. There exist two distinct approaches: the first is *Automatic Repeat reQuest (ARQ)* and the second is *Forward Error Correction (FEC).*

3.2 Automatic Repeat Request (ARQ)

In ARQ, only error detection methods are provided and no correction attempts are performed to recover packets in error. Instead, the receiver asks the transmitter to transmit the same data that were received in error. ARQ roughly works as follows: the sender keeps a record of the transmitted packets after transmission. The receiver acknowledges each packet received successfully by sending an ACK packet back to the sender. Packets that have not been successfully acknowledged in a predetermined time interval (i.e. packets that have *timed out*) are assumed to be lost, and hence they are retransmitted back to receiver. In some cases, the receiver, pointing the packets that need retransmission, transmits *negative acknowledgement (NAK).* Hence, unless the sender receives a NAK packet, no retransmission of any packet is needed.

The three most used ARQ protocols are the following: [7]

a) Stop and Wait (SW): in this algorithm, the receiver does not transmit any new packet unless all the previous packets sent are acknowledged. Meaning that the sender sends a packet and then waits until ACK is received. Once ACK is received, the next packet is sent. However, if the timeout expires and no ACK is received, the packet is declared lost and it is retransmitted. This method has a basic disadvantage. Since the transmitter does not use the channel allowed all the time, SW sets a limit on maximum data transfer rates.

b) Selective Repeat (SR): this algorithm is very similar to SW. The receiver acknowledges every packet that is received successfully. If one ACK was not received by the sender or it has timed out, the sender retransmits the lost packet and then continue sending fresh packets from where it left off. Since in SR algorithms packets are sent continuously, the inefficiency of SW is eliminated and the channel is used during almost all the session time. SR is usually implemented in applications where the receiver can accept out-of-order packets.

c) *Go-Back-N (GBN):* this algorithm tries to combine the benefits of both the previous method, SW and SR. like in SR, data is transmitted continuously. However, the receiver does not accept out-of-order packets. Meaning, when a packet is to be retransmitted by the receiver (due to the timeout of its ACK), all the following packets that are yet to be acknowledged by the receiver have to be sent also in sequence after the first packet. In this algorithm, packets are transmitted continuously, as in SR, without the need to buffer out of order packets and there is no re-sequencing overhead. All the previous ARQ techniques are shown clearly in Figures 8,9,10.

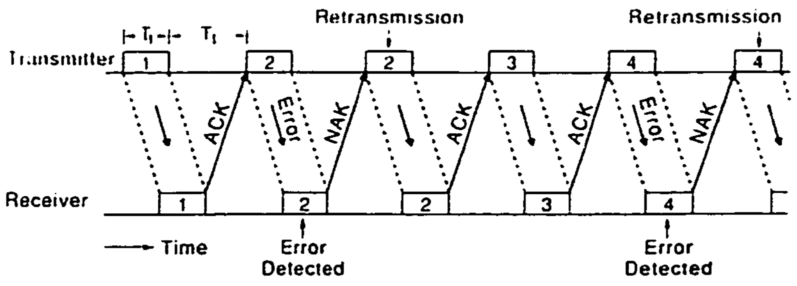


Figure 8: Example of Stop and Wait ARQ algorithm [11]

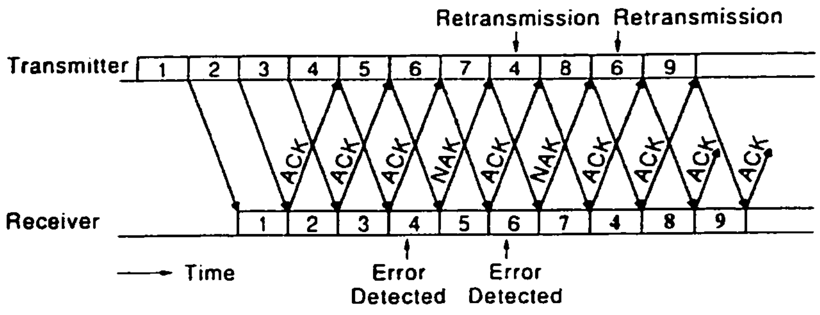


Figure 9: Example of Selective Repeat ARQ algorithm [11]

Despite the cost effectiveness of ARQ algorithms in many cases, ARQ has a major disadvantage; the throughput depends mainly on channel conditions. The worse the channel conditions are, the more retransmission of already-sent packets occurs. This may not be a serious problem in small systems and limited -number -of -users network, but it could have, however, serious drawbacks in big networks, like multicast sessions, if the error rates were high. Retransmission usually introduces a time delay because of the accumulation of round trip time of the ACK/NAK and retransmitted packets with the time out limits. This delay may not be acceptable for some application, especially interactive multimedia applications like video conferencing. Moreover, should the channel conditions become very bad, an implosion of ACK/NAK/ARQ packets could take place and the medium could be flooded by unrelated requests for packet retransmission and the QoS would noticeably degrade. It should be noted that implementing any type of ARQ requires the existence of a return path from the receiver to the sender, which may not be suitable for some systems like in mobile devices since feedback means consuming more power from the end user.

These restrictions on using ARQ mechanisms have led to the introduction of FEC methods, which can easily eliminate the drawbacks of ARQ.

3.3 Forward Error Correction (FEC)

FEC involves the addition of redundant data units (packets, bytes, or even bits) to be used to detect and correct any bits received in error, without any interaction with the sender; FEC is a mean to get it right from the first time [8]. Note that no return path from the receiver to the sender is needed to exist. FEC algorithms, just like ARQ, involve the addition of certain overhead that would decrease the throughput for a certain allowed network bandwidth.

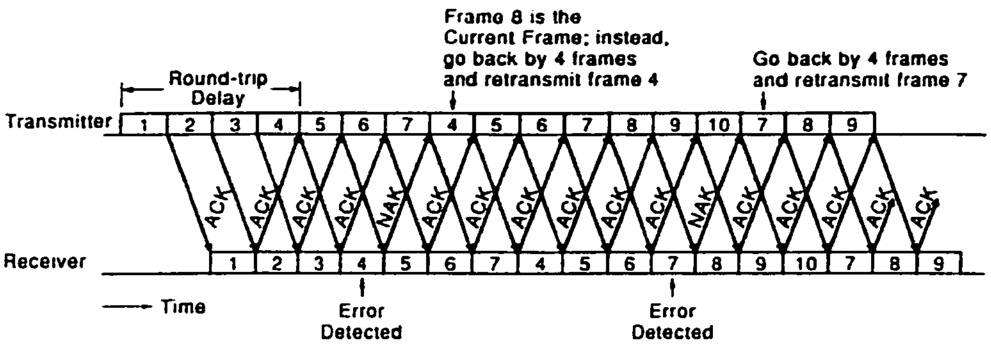


Figure 10: Example of Go-Back-N ARQ algorithm [11]

Forward Error Correction codes come in diverse forms. Two widely used categories of FEC techniques are *convolutional codes* and *block codes.* A quick overview of convolutional codes is presented, while more attention is given to block codes, since Reed-Solomon codes (one type of block codes) are the major interest of this paper and its real implementation. No detailed explanation of the mathematical approaches for either code is given, since these explanations are beyond the scope of this thesis.

3.3.1 Convolutional Codes

Convolutional codes are a popular class of codes that have memory; the coding of one block of data would depend, in addition to the input stream and the generator polynomial, on previous coded blocks. Convolutional codes are usually described by *(n,k,m)* combination[7]; a convolutional code generates *n* encoded bits for every *k* information bits with a memory that extends *m* stages. Figure 11 shows a (2,1,4) convolutional encoder. Every 1 input bit generates 2 output bits, which depend on the previous 4 input bits. The encoding equations ruling this example are:

$$V^{(i)}(D) = U(D)G^{(i)}(D), \qquad i = 1,2$$
where
$$U(D) = u_0 + u_1 D + u_2 D^2 + \ldots$$
and
$$G^{(1)}(D) = 1 + D + D^4$$
$$G^{(2)}(D) = 1 + D^2 + D^3 + D^4$$

where $G^{(i)}(D)$ is the ith generator polynomial of the code [7].

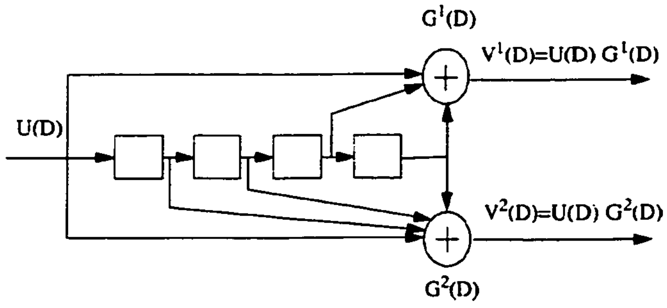


Figure 11: Rate 1/2 convolutional encoder with m=4 [7]

Convolutional codes are widely implemented since they are very easy to implement and they only involve simple arithmetic operations. Decoding of these codes is usually done by Viterbi algorithms, and the decoder should have a previous knowledge of the history of the decoded stream [7]. If the decoder, for any reason, lost track or made a mistake in the history of the stream, the error would propagate. It should be noted that a large number of operations per bit is required for decoding, and the complexity of decoding increases as m increases. The differences between convolutional codes and other types of coding are stated in the next section.

3.3.2 Block Codes

Block codes differ from convolutional codes in that they divide the bitstreams into non-overlapping block of data, and then these blocks of data are encoded separately and independently from other blocks. The main class of block codes used today is the *linear cyclic* [7], since they are easy to implement. The key idea of these block codes is to take k blocks of data and encode them to become n blocks (where $n>k$) in such a way that any subset of k encoded blocks suffices to reconstruct the source data. Figure 12 shows this idea clearly.

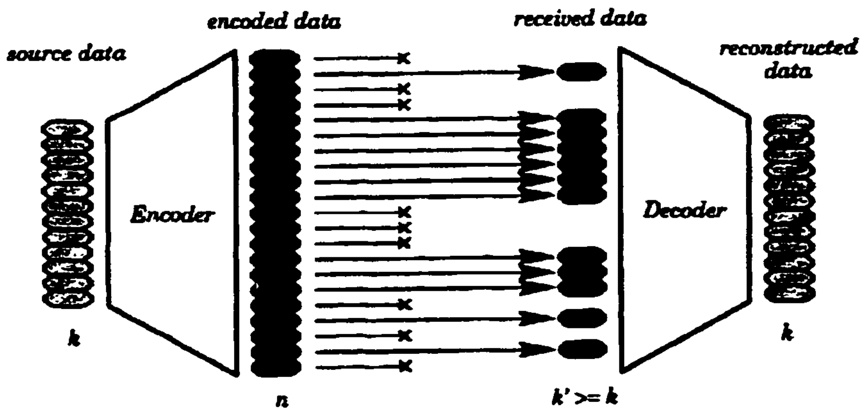


Figure 12: Graphical representation of block encoding/decoding [9]

Linear codes are easy to implement since they involve linear algebra calculations. Assuming that $X = [x_0 \quad x_1 \quad x_2 .. \quad x_{k-1}]^T$ is the $kx1$ input matrix and G

is the *generator matrix*, an *n x k* well chosen matrix, then the output (encoded data) is [9]:

$$Y = G \times X$$

where *Y* is an *n x 1* matrix that represent the encoded data. It should be noted that to be able to decode the incoming data at the receiver for any subset k, the decoder should be able to reconstruct at least *k* linear independent equations. This holds if any $k \times k$ matrix extracted from G is invertible [9].

The two most common linear cyclic block codes are BCH and Reed-Solomon codes.

3.3.2.1 Bose-Chaudhuri-Hocquenghem (BCH) Codes

This code was developed between 1959-1960. BCH are binary codes, meaning that the block unit in this code is the *bit*. BCH codes are defined by their generator polynomial *g(D)* in any given finite Galois Field of degree *m* (GF (2^m)). Since a property of algebra states that for any m, there is at least one primitive polynomial, it can be stated that for any m $(m \geq 3)$ and $t < \frac{N}{2}$, there exists a binary t-error-correcting BCH code with $N=(2^m-1)$, $N\text{-}K \leq mt(K \geq 2^m - 1 - mt)$ and $d_{min}=2t+1$ [11]. The generator polynomial can be calculated by taking the least common multiple of all the minimal polynomials of the field elements $GF(2^m)$. The coefficients of the generator polynomial belong to the basic field GF(2). [11] shows the parameters and generator polynomials for some BCH codes taken for different values of *m*.

3.3.2.2 Reed-Solomon (RS) Codes

Reed Solomon codes are special subclass of BCH code where the blocks are not composed of binary bits, instead they use *symbols*. This symbol can be of any number of bits. The basic difference between binary and non-binary codes lies in the decoding process; non-binary decoders must be able to find the error location and error magnitude, while in binary codes it is enough to find the location of the error. In BCH codes, the symbols (1 or 0) of the codes lie in GF(2). On the other hand, the values of the symbols of RS lies in GF(q^p), where q is any prime number and p is any integer power of q. In RS, both the base field and the extension field are the same, meaning that both the coefficients of the generator polynomial and RS symbols belong to the same GF(q^p). The most common value for q is 2 in RS codes. The generator polynomial of t-error-correcting RS code in GF(2^p) is the least degree polynomial that has $\beta, \beta^2, \beta^3 ... \beta^{2t-1}$ as roots, where β belongs to GF(2^p). The minimal polynomial over GF(2^p) of an element β in GF(2^p) is the factor $m_\beta = x + \beta$ [12]. The generator polynomial of Reed Solomon code is:

$$g(x) = (x+\beta)(x+\beta^2)....(x+\beta^{2t})$$

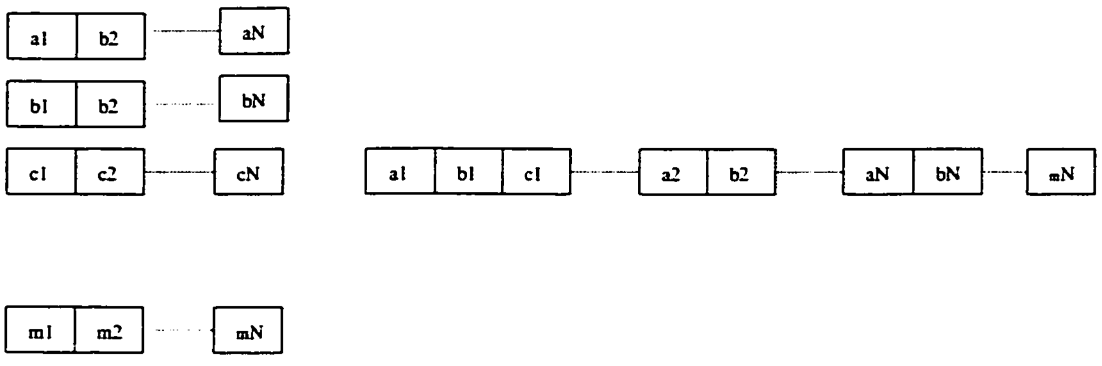
The coefficients of g(x) are not anymore binary, as in BCH codes, instead they belong to the GF(2^p).

The code takes k data symbols of length p bits and gives n data symbols with the ability of correcting n-k symbol erasures and $(n$-$k)/2$ symbol errors. That is because, the decoder has n-k data symbols to spend on decoding. If the place of an error were unknown, the decoder would spend one redundancy to locate the error and one redundancy to recover it. If, however, the decoder could be informed beforehand about the locations of the errors, it would spare it one redundancy [8]. The most common value of p is 8. This means that the symbol consists of 1 byte, which is very suitable for many applications including the implementation over MPEG-2.

The most common type of RS codes is *systematic codes.* In systematic coding, the first K symbols of the N encoded RS word would be the original data. The remaining N-K symbols would be the parity symbols. Systematic codes are much easier to manipulate than non-systematic ones. RS codes are very efficient and powerful codes. RS codes are efficient against random and burst errors and they have a very low misdecode rate and this decode error rate is usually 5 orders of magnitude below the Bit Error Rate (BER) [8]. Furthermore, since RS codes work with symbols, they can be multiplexed and manipulated. Finally, RS codes are very suitable for high rate data transfer since there exists RS decoders which can accommodate up to 120 Mbps decoding rate, and RS decoders which can faster than 2 Gbps are now investigated [8].

3.4 Interleaving

It is a common approach to use interleaving along with FEC and especially RS codes. Interleaving is a typical way to spread errors; it involves rearranging burst errors to make them random. The technique of interleaving is simple, as Figure 13 shows; m codewords, each of data length equals to N, are written row-by-row into an m x N matrix. However, when a packet is to be constructed, the data is read out column-by-column and then is sent to a receiver.



Before interleaving

Bitstream after interleaving

Figure 13: Interleaver

The major role of the interleaver is to decrease the number of errors in each code word. However, it is not very much efficient against long term burst errors. [7] states that the FEC and interleaving strategy is effective when *tm* exceeds $1/r$, where t is the correction capability, m is the *interleaving depth*, and *r* is the average burst length. It should be noted that using interleaving methods introduces delay at the receiver, since the receiver should wait for a complete interleaved window to reach in order to obtain meaningful information.

3.5 Comparison between Convolutional and Block Codes

In the following table, some of the differences in usage of convolutional codes and block codes are shown. In some cases, both of these codes are implemented at the same time in one system to get what would be called *concatenated codes*. Figure 14 shows a concatenated system. The outer encoders/decoders are chosen to be RS, while the inner encoders/decoders are chosen to be convolutional. The inner decoders' role is to reduce poor quality data to medium quality data, and the outer ones reduce medium quality data to very good data [8].

Convolutional Codes	***Block Codes***
Best suited in Gaussian noise environment, less efficient with burst errors	They are superior against burst errors and random
High complexity in terms of decoding operations per bit. Real numbers addition and multiplication	Lower number of operations per symbol. Operate directly on bits and real operations are avoided
Decoder complexity increases as redundancy decreases	Decoder complexity decreases as redundancy decreases
Best suited for low- speed -rates decoding	Able to handle decoding high speed rates

Table 4. Differences between convolutional codes and block codes

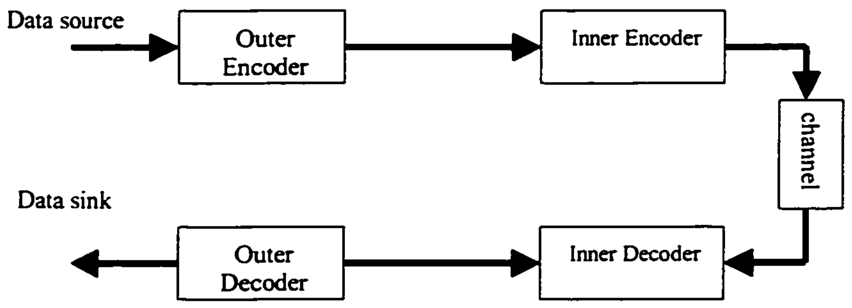


Figure 14: A concatenated system

Finally, a major disadvantage in FEC methods, whether convolutional or block codes, is that they introduce a significant overhead. In cases of a channel with a near-zero error rate, this overhead would exhaust the networks resources and bandwidth without being necessary.

3.6 Hybrid ARQ/FEC Techniques

ARQ, as explained, is simple and efficient when error rates are not large. However, it leads to variable delays and its throughput is directly dependant on the channel conditions. FEC schemes maintain constant throughput independently from channel conditions but with high overhead. Furthermore, FEC methods are stationary and they may not be totally adequate to implement in non-stationary channels. These properties complementary properties of ARQ and FEC immediately suggest the combination of the two schemes in one and it is called *hybrid ARQ/FEC*. Hybrid ARQ/FEC method is divided into two types: Type -I Hybrid and Type-II Hybrid.

Type I Hybrid behaves as follows: the data are FEC encoded and sent through the network; both the data and the parity bytes are sent. If the packets are received by the end point with no error or with some errors (less than N-K), the original data is decoded by FEC decoder and recovered. If the number of errors exceeds the capability of the decoder, the receiver discards the packet and asks for retransmission of the same packet another time. This Type-I Hybrid was found to be most efficient in a uniformly noisy channel that is exposed to frequent bursts [8]. The problem of this type is that redundancy exists even when no errors exist.

Type II Hybrid is slightly different than Type I. In Type-II, the data is sent without the FEC parity bytes, but some error detection codes are added. If the data is received error-free, there is no need for the parity bytes. If errors are detected, the receiver sends a request to the sender. In this case, the transmitter does not send the same data another time, instead the FEC parity bytes are sent and the decoder will try to decode the packet in error to come up with the original data. In [12], some changes were made to this Type-II method. The data are encoded using two codes, C_1 and C_2

to come up with a pair of code words, c_1 and c_2. These codes can decode and recover the original data when performing D_1 and D_2 decoding operations. c_1 is first transmitted and c_2 is set aside. If the receiver could not decode the word, it asks for c_2 and then tries to decode the data performing the process D_2. If the decoding process fails again, c_1 and c_2 are combined to get c_3 and it is sent again to the receiver to perform D_3 decoding process. This process is repeated until the data is decoded perfectly.

The major focus in the chapters to come will be on layered Hybrid ARQ/FEC method that relies on Hybrid Type-I principles.

CHAPTER 4

Implementing Layered Hybrid ARQ/FEC on MPEG

4.1 Objective

The aim of this thesis is to implement a piece of application-level software that tries to exploit the properties of MPEG video. in order to achieve the best QoS at the time of videos' transmission. Generally speaking. QoS of MPEG increases as the throughput increases and as errors and erasures decrease. The approach proposed in this thesis in order to decrease the percentage of errors and erasures is based on Layered Hybrid ARQ/FEC algorithm. It is worth noting that the software that was designed for this purpose was implemented and tested in the laboratories of ETS (École Superieur de Technologie)-Montreal- Canada. Data was taken and analysed and then conclusions were made accordingly. as it will be clearly discussed in the rest of the thesis.

4.2 Introduction to the Proposed Approach

It was mentioned earlier that the degradation in QoS due to errors and packet erasures is directly related to *where* these errors hit and in what type of frame, and Figure 7 clearly shows that. Errors in I Frames have the most effect on the QoS of the MPEG stream. Errors in P Frames have less effect and they have the least effect in B Frames on the overall QoS. This property of MPEG can be exploited during FEC encoding videos- to- be- transmitted via lossy channels; different levels of FEC protection can be given to different types of Frames. I, P, and B frames are to be

(n,k_I), (n,k_P), and (n,k_B) RS encoded, respectively, where $k_I < k_P < k_B$. In addition to layered RS encoding, the approach proposes to integrate this method with SR- ARQ to get a Type I Hybrid ARQ/FEC algorithm. The SR-ARQ proposed here is slightly different of the one shown in Figure 9. No ACKs are sent from the receiver to the sender. Only NAK packets are to be communicated end-to-end.

The algorithm of layered FEC encoding was proposed earlier in [13]. In [13], actual experiments were conducted to show the different effects of errors on different types of packets. MPEG video streams were separated into 3 layers; I, P, and B layers as Figure 15 suggests.

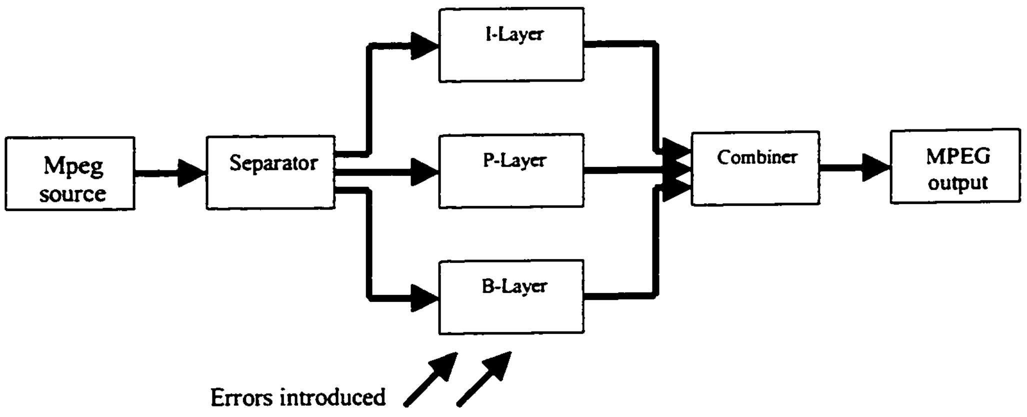


Figure 15: Separation of MPEG video into layers [13]

The following experiment was conducted in [13]. Errors that have a Poisson distribution with a range of mean error rates were introduced to each layer in turn, and then to the whole MPEG sequence. The MPEG video that was tested was a 500 frames 320x240 sequence, and its GOP parameters were N= 10, M= 5. The target was to show that I Frames are not as tolerant to errors as P and B frames. The Mean Square Error (MSE) for the *whole* decoded stream, in addition to subjective evaluation of picture quality, were calculated for different ranges of error values. The results were as expected; the MSE of the video stream is highest when only I Frames are destroyed and it is lowest when only B frames are destroyed. Figure 16 shows the results of the experiments done in [13].

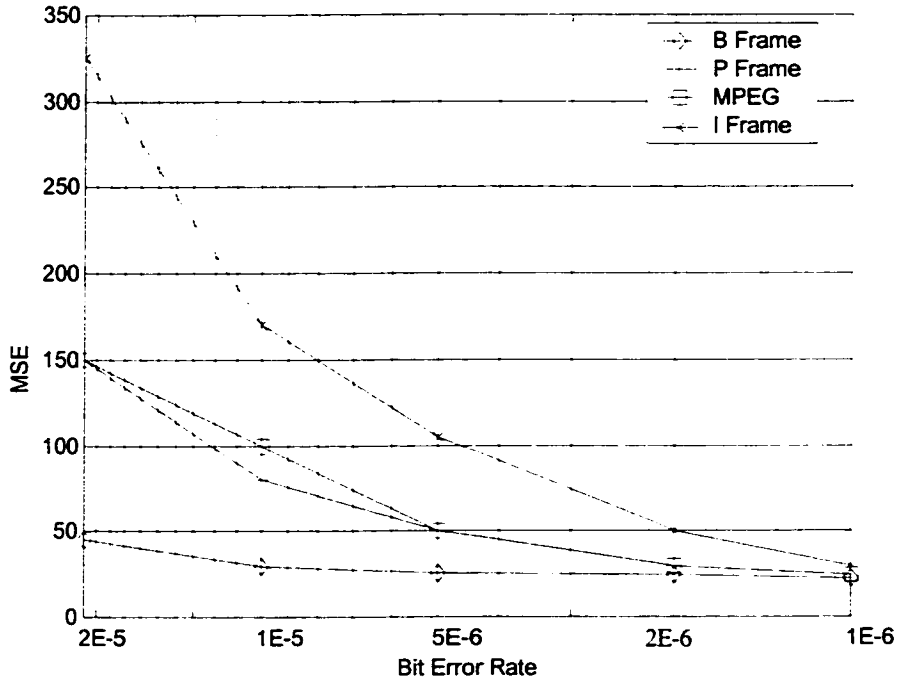


Figure 16: MSE values for test sequences [13]

To give more emphasis to this specific characteristic of MPEG video, two small tests were conducted in the labs of ETS on the MPEG stream Miss America (whose specifications are shown in Table 5). In the first test, four consecutive bytes were destroyed in the first I Frame of the sequence. The result was that the error in the slice propagated for 9 frames (the number of frames in the GOP) before the image becomes clear again, as it can be clearly shown from Figure 18. In the second test, eight consecutive bytes were destroyed in the first B Frame of the sequence. The result was that no noticeable effect was detected on the quality of the sequence.

It is worth mentioning that in [14], layered FEC for MPEG-1 was introduced *theoretically*. The method of giving different priority to different types of frames was name as *Priority Encoding Transmission (PET)*. The main idea of PET is to assign redundancy to a GOP, where this redundancy is not evenly distributed among the frames of the MPEG sequence. As Figure 17 shows, the basics of PET can be briefly explained as follows: the GOP is encoded into n packets with a total size of all these n packets is N Kbytes. The mapping of the GOP is done in a way that some data from every frame is contained in each of the packets (sort of scrambling). This will lead to more robustness in the presence of bursty errors [14]. The redundancy could be created either by using Galois Fields, or by using Cauchy Matrices. The other important focus of [14] was to calculate the optimum partitioning of the redundancy among the frames, so that the overhead would be minimal. Some referring to this method will come in the next chapter.

4.3 Algorithm of the Proposed Approach

This section explains the proposed approach (Layered Hybrid ARQ/FEC). This new approach tries to give more robustness for MPEG video bitstream during communication sessions, using layered (*or differential*) Hybrid ARQ/FEC. Meaning that each type of frame in the MPEG bitstream will be treated differently, according to its level of importance. This approach is to be implemented as client/server software, on C language platform. Figure 19 shows a flow graph of the proposed technique.

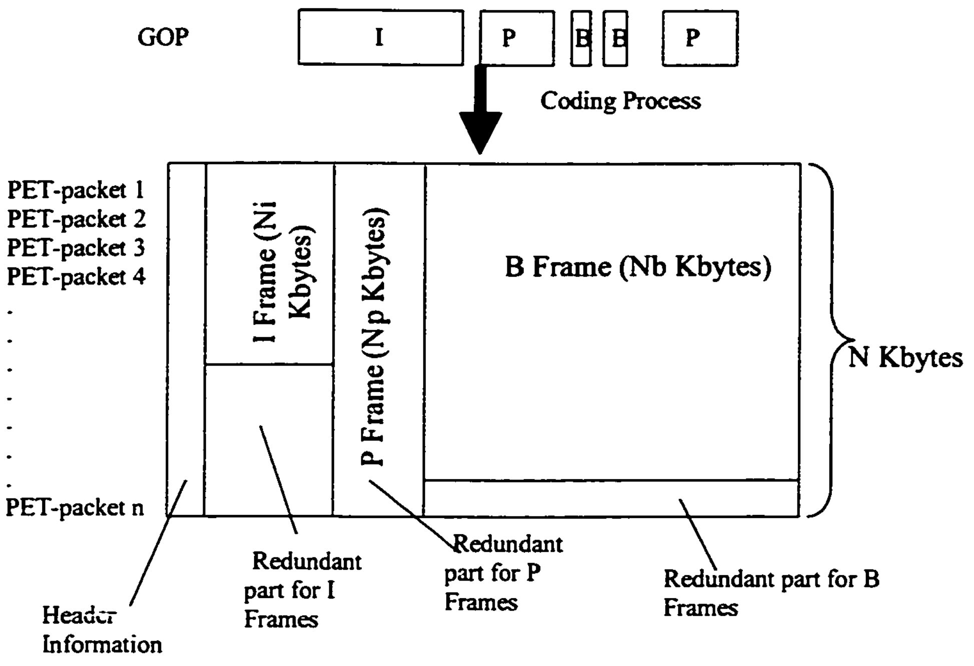


Figure 17: Illustration of the coding process employed in PET [14]

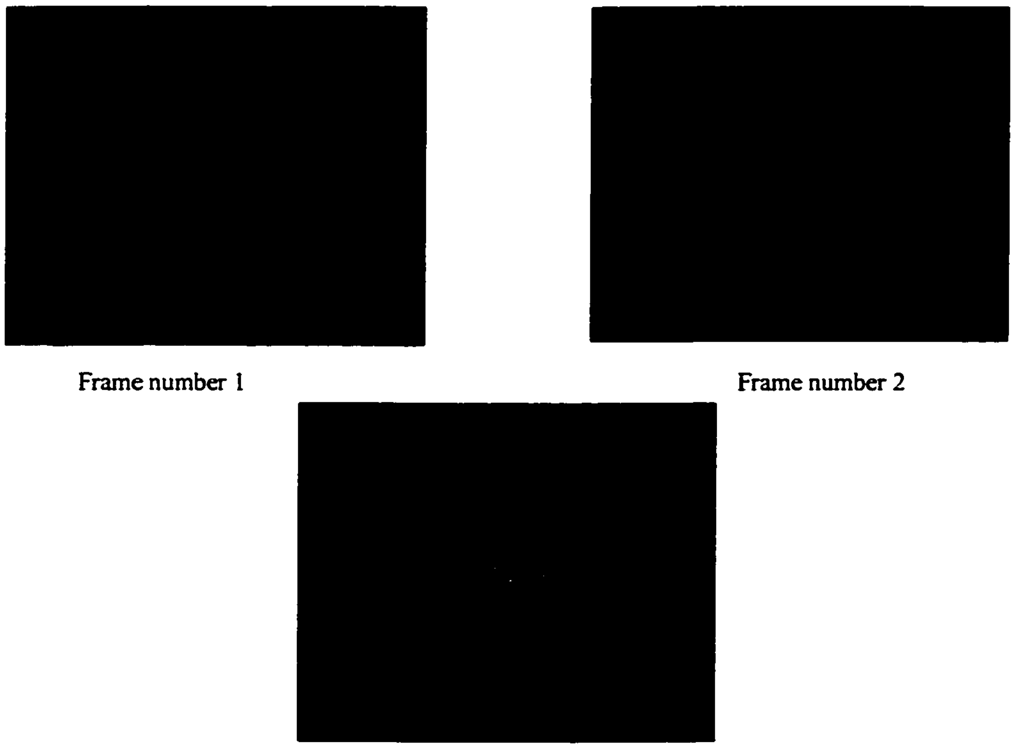


Figure 18: The effect of destroying 4 Bytes of an I Frame in Miss America Sequence

The Specifications of each stage, which is used in the practical implementation, are explained in the sections to follow.

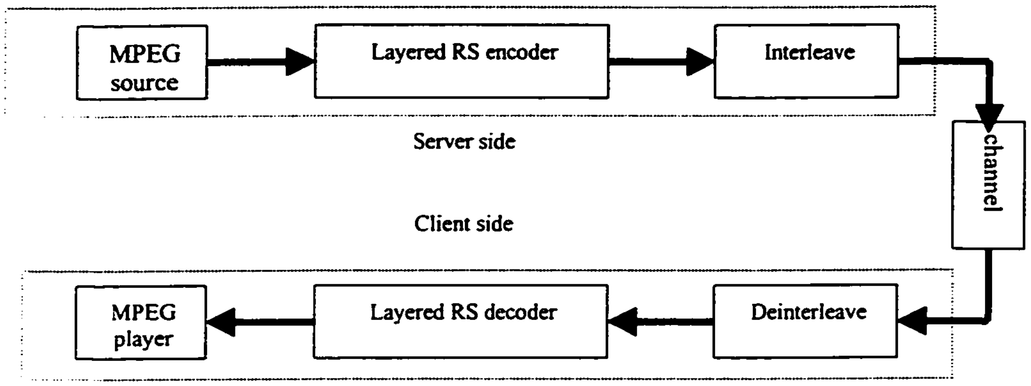


Figure 19: Flow graph of proposed technique

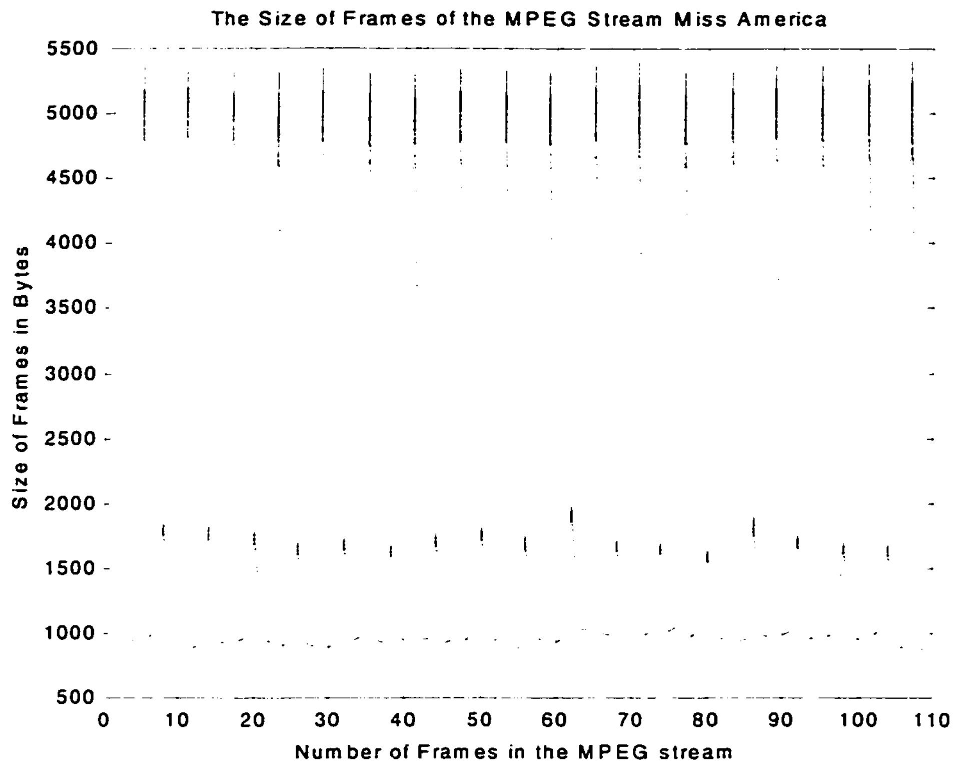
4.3.1 MPEG Source

Two different MPEG bitstreams are tested in this experiment. Their specifications are shown in the following table.

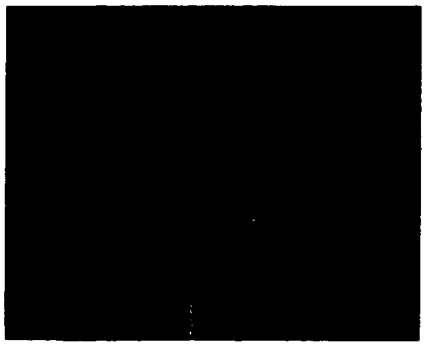
Specification	***Miss America***	***Flower Garden***
Resolution	352 x 288	352 x 240
GOP	I B B P B B	I B B P B B P B B
% I Bytes	52.05	35.35
% P Bytes	16.35	55.74
% B Bytes	31.60	8.91
Number of Frames	109	60
Bit Rate (Mbps)	0.5	1.5
Properties	Black and white, slow motion, high interframe correlation	Colored, moderate motion, also high interframe correlation

Table 5: Specifications of tested MPEG streams

The Size of Frames and some error-free shots of the two MPEG bitstreams are shown in Figure 20 and 21.

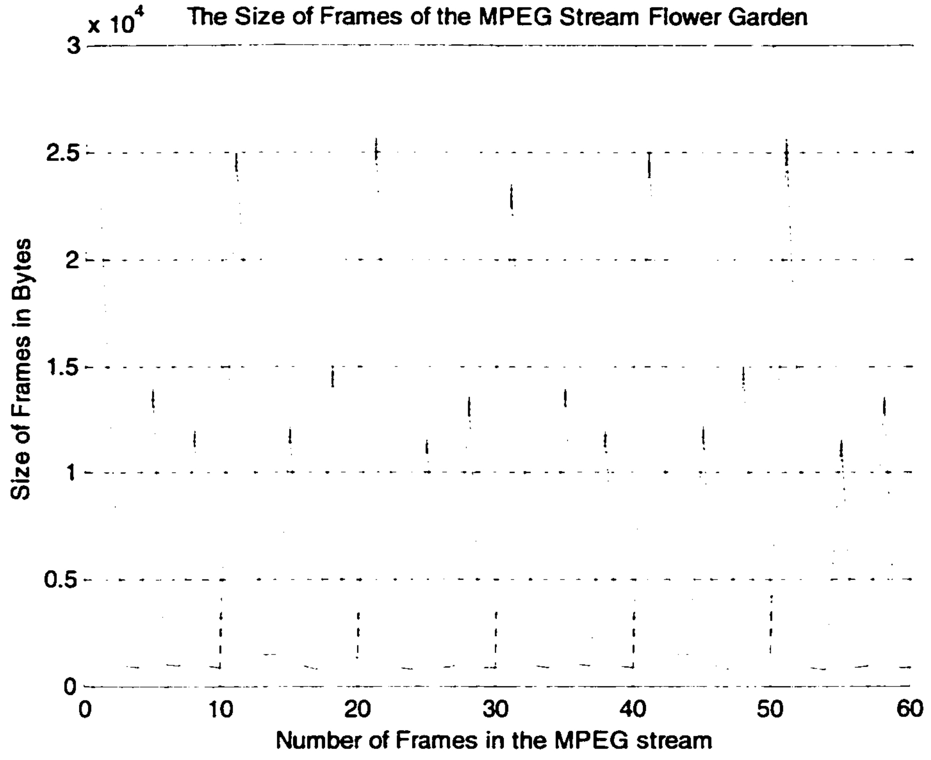


Frame number 95


Frame number 96

Frame number 98

Figure 20: Miss America , frames' sizes and screen shots



Frame number 2

Frame number 3

Frame number 7

Figure 21: Flower Garden, frames' sizes and screen shots

4.3.2 Layered RS Encoding/Decoding Specifications

The systematic Reed-Solomon encoder used in the experiments done for this thesis was (255,k), where k is the number of bytes (since the number of bits in the symbol is set to 8), and it is different for each type of frame. Three values of k are chosen; 191 bytes for I Frames and system headers, 207 bytes for P Frames, and 223 bytes B Frames. Meaning that the encoder at the server side will take 191, 207, and 223 bytes from I, P and B Frames, respectively, and add 64, 48, and 32 parity symbols to them to produce 255 -byte packets. It should be noted that these are the main values of k that will be used; however, sometimes during the experiments these values will be changed to observe the effects.

With the values stated above, the encoded RS words for I, P, and B Frames can tolerate 64 erasures/32 errors, 48 erasures/24 errors, and 32 erasure/16 errors, respectively, and the client-side RS decoder would still be able to decode successfully. The reason that enables the RS decoder to fix more erasures than errors is explained the previous chapter in section 3.3.2.2.

4.3.3 Interleaver/Deinterleaver

The Interleaver used throughout the experiment, unless stated otherwise, has a depth of 256 words. Meaning that the interleaver will take 256 words and each one of them has 256 bytes and will interleave them as in Figure 13, to produce 256 interleaved words. On the other hand at the client side, the deinterleaver will rearrange the bytes in order to get the original 256 words (not interleaved).

4.3.4 Procedures at the Server Side

The server role is divided into two main parts; 1) encoding the MPEG file (layered RS encoding and interleaving) and 2) sending the data and processing NAK requests.

4.3.4.1 Encoding Process

The first process is shown in general in Figure 25, and is explained in details in Figure 26. The program performs byte-level search for the system and picture headers and locates them. The values of these start codes and headers can be taken from Table 3. Each located frame is cut into packets containing number of bytes (185 bytes for I Frames and system information, 201 bytes for P Frames, and 217 Bytes for B Frames), as shown in Figure 22, and then 6 bytes are added to each packet. The 4 bytes that create the header are one byte for the Type (to indicate whether the packet is I, P or B), two bytes for Sequence numbering, and one byte for Data Length (DL). DL byte is an indication of how many *dummy bytes (padding)* exist within the data part of the packet. Byte stuffing is performed whenever data bytes are less than the above-mentioned numbers. The 2 remaining bytes are added at the end of the packet and they are *Cyclic Redundancy Check (CRC)*. CRC performs error detection at the receiver and it protects the data and header bytes.

The second phase of the encoding process is the addition of RS parity bytes. As Figure 23 shows, 64, 48, and 32 parity bytes are added to I, P, and B packets respectively, and all the RS encoded packets now have the same length, 255 bytes. At the end of this systematic RS encoding, one dummy byte is added at the end of the RS packet. The only task of this byte is to make the packet of length 256 bytes, the thing that facilitates all upcoming operations in later stages.

The last phase is to interleave each 256 packets together; i.e. with a depth of 256. The first interleaved word will contain bytes number 1 of all the 256 RS encoded words, the second interleaved word will contain bytes number 2 of all the RS encoded words, etc.... The performance of the interleaver depth will be evaluated later in the thesis. Following interleaving, another 3 header bytes are added to all interleaved packets: one byte for Type (whether the packet is a new one or a retransmission), and 2 bytes for Sequence numbering. That means that every interleaved word to be sent is of length 259 bytes.

It should be mentioned that all the phases describe above are done *offline* since they are time consuming and do not add anything to the algorithm this thesis is trying to implement.

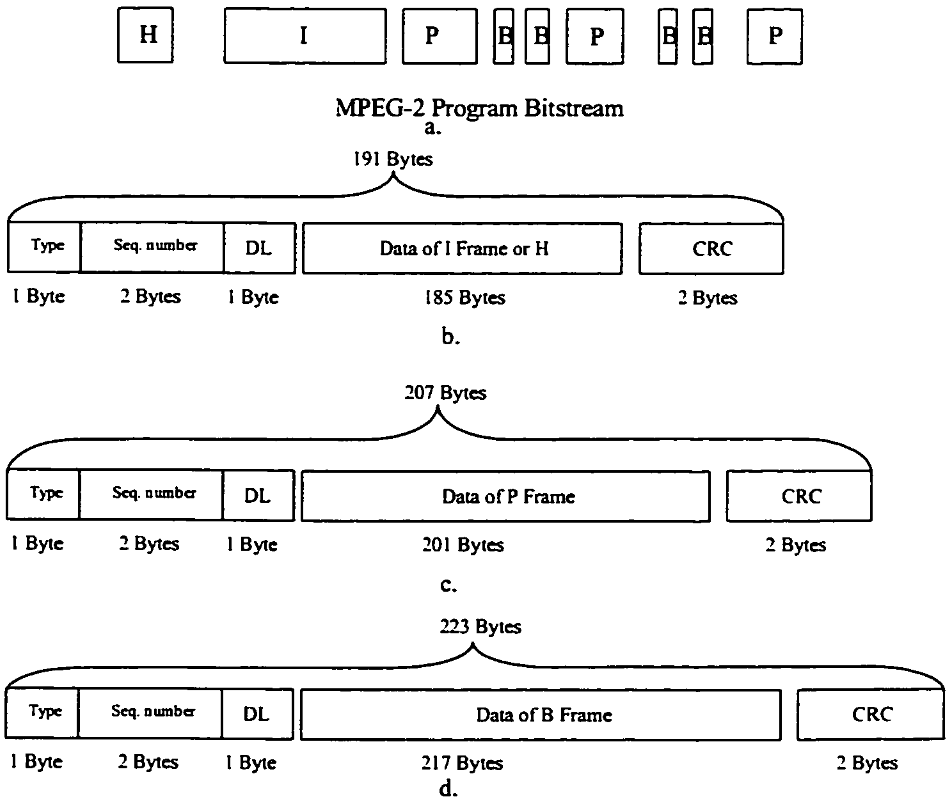


Figure 22: a.) MPEG stream with different types of frames, then adding headers and CRC to I Frames, P Frames, and B Frames in b.), c.), and d.), respectively

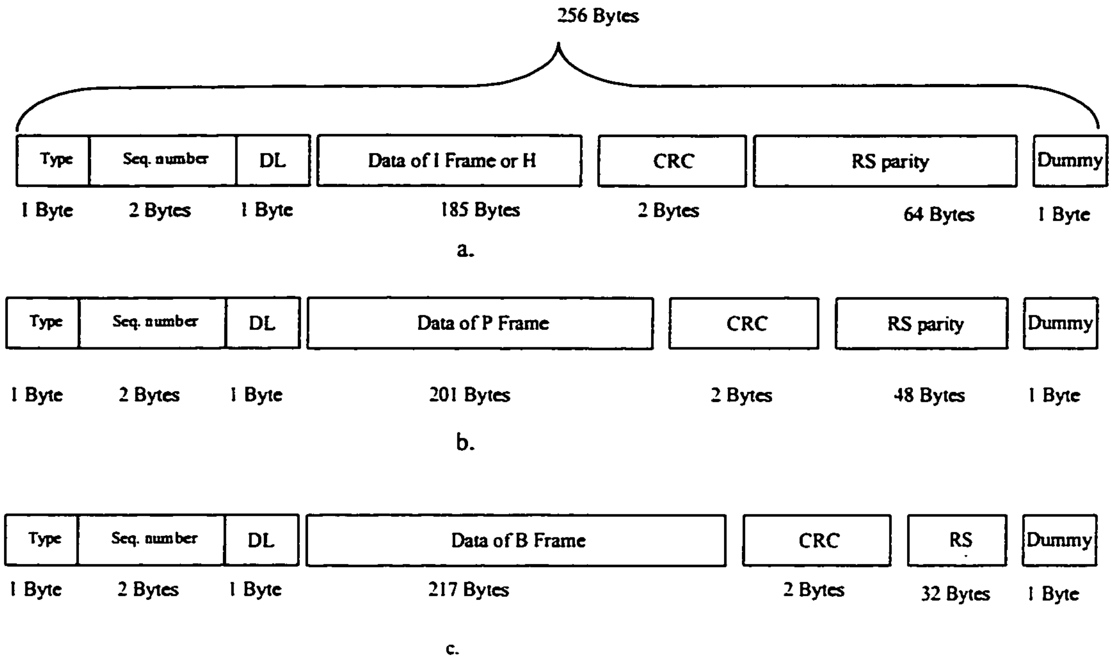


Figure 23: Systematic Reed-Solomon Encoding and adding the dummy byte for I, P, and B packets in a.), b.), and c.) respectively

4.3.4.2 Transmitting the Packets and Processing ARQ

When the encoding process finishes, the server starts sending the interleaved packets to the receiver. The server loads a block of interleaved packets into the memory, and the number of words in the block equals the interleaving depth (256 in this case). The block that is prepared to be sent is called *current window*. Then, the server combines several packets together and sends them to the UDP layer so they can be delivered to the final destination. In this program, UDP protocol, not TCP, is used as the transport protocol, since UDP is connectionless and introduces lower overhead and faster communication. The lower layers are not of concern in this thesis and the application implemented does not give any assumption about them. As soon as all the packets in the current window are sent, the server moves a copy of them to another place in the memory, and now this block is called *previous window*, and then loads another fresh block in place of the block sent, and this block is now called current. Saving a copy in the previous window is necessary since it will be used to process one type of ARQ requests that are coming from the receiver.

In the method proposed, the receiver could send two kinds of NAKs to the server: *Fresh NAKs*, which requests non-interleaved packets, and *Interleaved NAKs*, which requests interleaved packets that were already sent. The retransmission of non-interleaved packets would save the client some processing, since no deinterleaving is performed. The existence of those two types of NAKs would require that the server load a copy of non-interleaved (RS encoded) previous window into memory along with previous interleaved window.

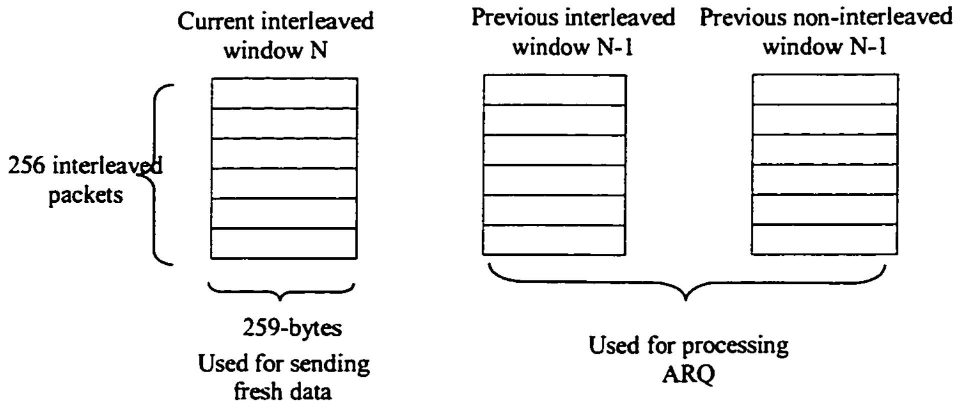

Current interleaved window N
Previous interleaved window N-1
Previous non-interleaved window N-1
256 interleaved packets
259-bytes
Used for sending fresh data
Used for processing ARQ

Figure 24: Memory arrangement in the server

The server implements Selective Repeat ARQ algorithm (SR-ARQ); it continues to send packets as long as no ARQ is received. As soon as a NAK is received, the server looks in the previous windows to determine whether the requested packets are still saved. If the packets are found, they are directly retransmitted to the receiver and then the server continues sending the fresh packets, and if they are not found, the request is dropped. The transmission and ARQ processing procedures are shown in Figures 27 and 28, respectively.

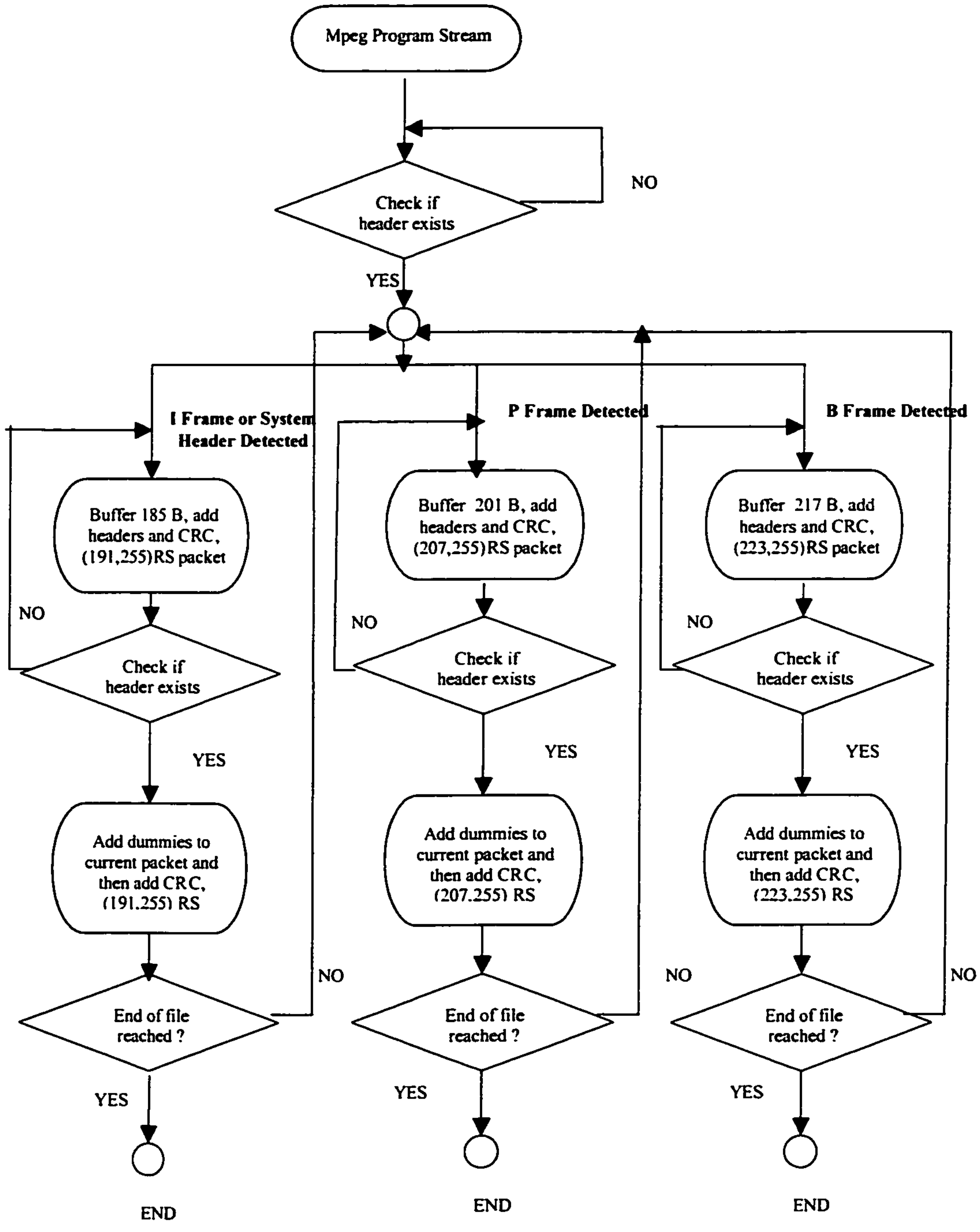

Mpeg Program Stream
Check if header exists
NO
YES
I Frame or System Header Detected
P Frame Detected
B Frame Detected
Buffer 185 B, add headers and CRC, (191,255)RS packet
Buffer 201 B, add headers and CRC, (207,255)RS packet
Buffer 217 B, add headers and CRC, (223,255)RS packet
Check if header exists
Add dummies to current packet and then add CRC, (191,255) RS
Add dummies to current packet and then add CRC, (207,255) RS
Add dummies to current packet and then add CRC, (223,255) RS
End of file reached ?
END

Figure 25: Flow Chart of the offline encoding of MPEG stream. Every type of frame is encoded differently as shown.

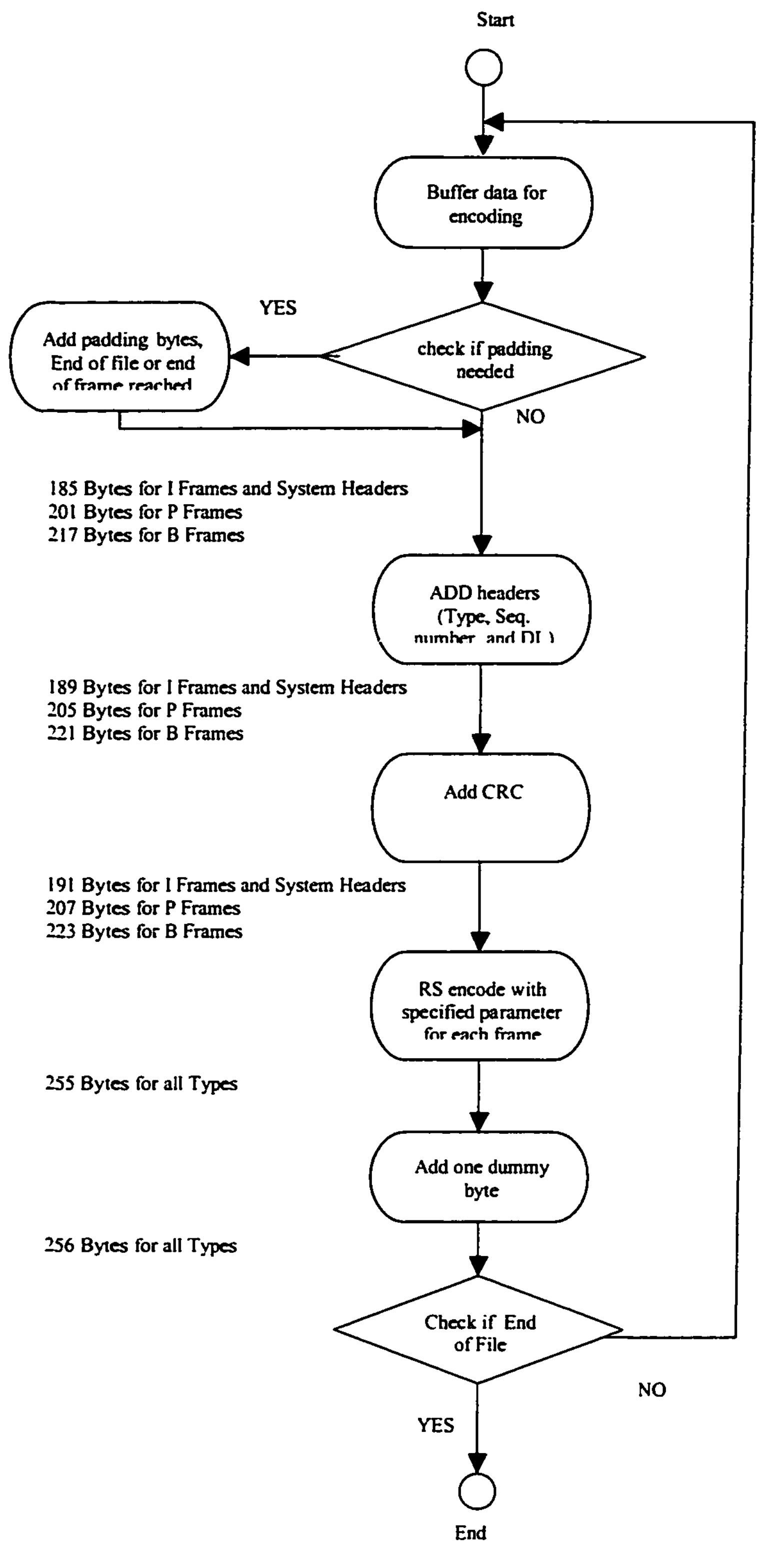


Figure 26: Flow Chart of the offline encoding process of MPEG stream in details

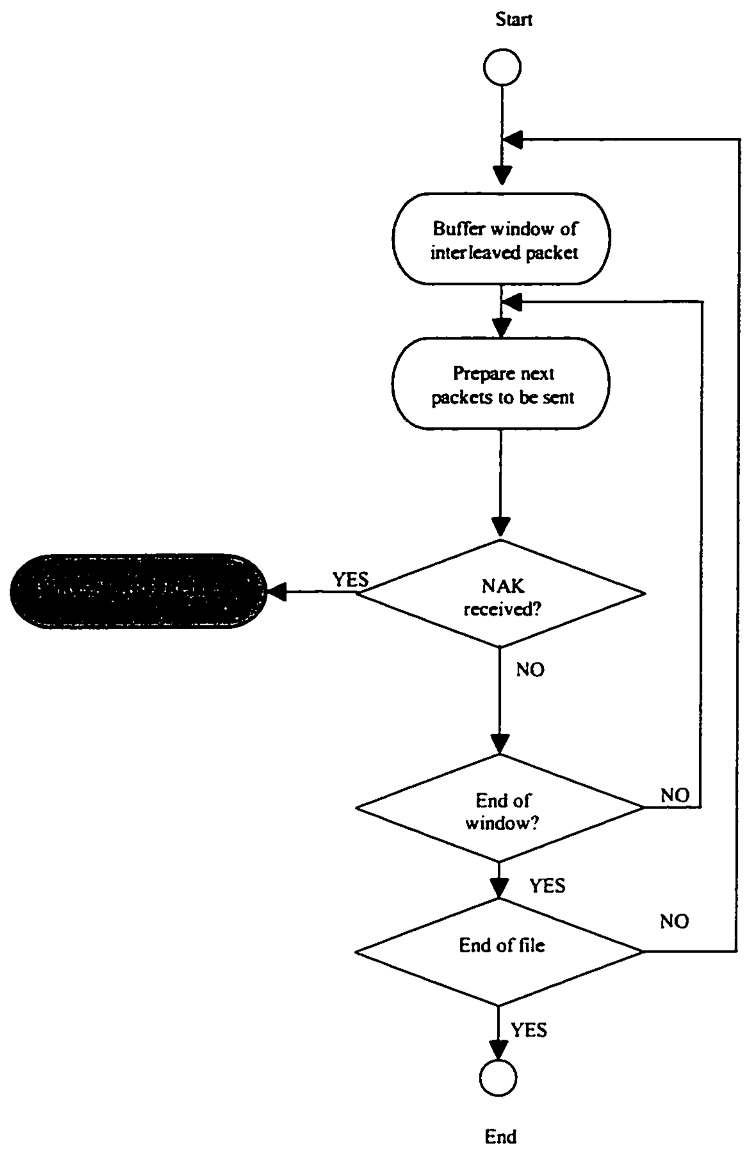

Start
Buffer window of interleaved packet
Prepare next packets to be sent
NAK received?
YES
NO
End of window?
NO
YES
End of file
NO
YES
End

Figure 27: Flow Chart of server program

4.3.5 Procedures at the Client Side

The client is assumed to be aware of all the crucial information such as k_I, k_P, k_B, interleaving depth, etc... The process at the client side can be divided into two parts: the first is processing the packets in previous window, and the second is filling the memory of the current window.

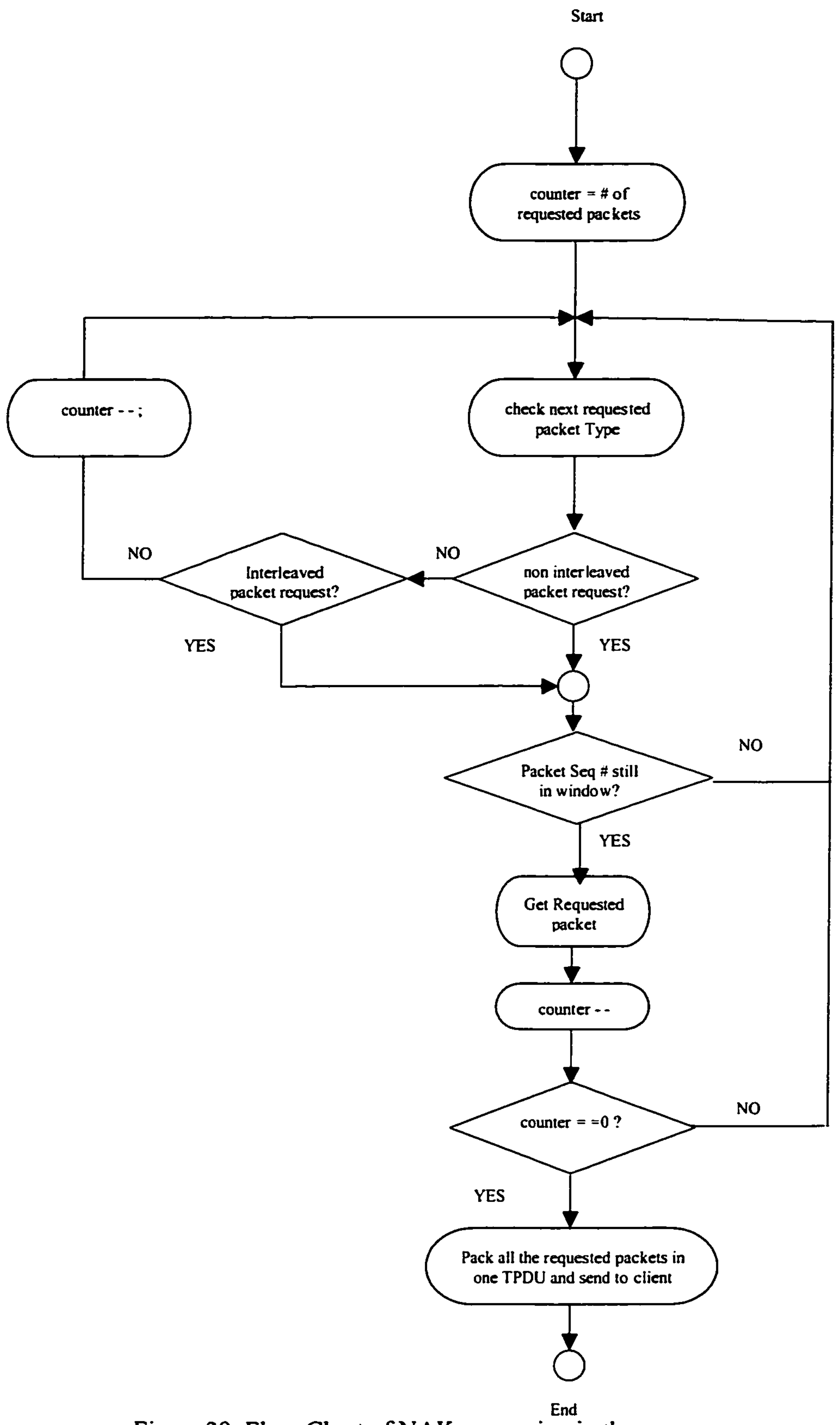

Start
counter = # of requested packets
counter - - ;
check next requested packet Type
NO
Interleaved packet request?
NO
non interleaved packet request?
YES
YES
NO
Packet Seq # still in window?
YES
Get Requested packet
counter - -
NO
counter = =0 ?
YES
Pack all the requested packets in one TPDU and send to client
End

Figure 28: Flow Chart of NAK processing in the server

4.3.5.1 Processing the Previous Window at the Client

The client has first to buffer one interleaved block (256 packets) and then it processes its packets. Meaning that the client will not start decoding the data unless the first packet of the next window is received. This will give the client some time to make sure that the maximum number of packets in the window has been reached. It first deinterleaves the packets with the aid of the Sequence number header, and then it reads sequentially the Type header of every RS encoded packets (deinterleaved). As soon as the type of packet is known, CRC is calculated and if it is correct, this packet will be delivered directly to the playing buffer. If, however, the CRC is not correct, the client starts RS decoding for *this* packet. If the decoding is successful, the data is delivered to the playing buffer, and if not the client sends a *Fresh NAK* asking the server to transmit a non-interleaved packet and a timer is set. If the client receives the requested word before the timer times out, the packet is delivered *unchecked* (no CRC and no RS decoding) to the playing buffer, otherwise the packet is delivered *as is*. The flow graph of this process is shown in Figure 29

4.3.5.2 Filling the Current Window

Filling of the current window is an *interrupting process* for the above-described one. This means that the client is *always* in the process of decoding unless some packets are detected in the UDP reception socket. When packets are detected in sockets, the client stops decoding of the previous window and starts filling the current window. First the Type header of the interleaved word is checked, to determine whether this packet is a retransmission or a new one and then Seq. Number is checked. If it is new and belongs to current window it is buffered and if it belongs to next window, the *need_to_flush Flag* is set. When this flag is set, the client delivers all the remaining packets in previous window (whether they have erasures/errors or not) to the playing buffer. If the word is a retransmission and it belongs to previous window, the packet is delivered unchecked to the playing buffer.

It is worth noting that the client tolerates out-of-order packets belonging to the same window, since it can reorder them according their sequence numbers. The filling process is described is Figure 30. The values of different Types of headers are given in Table 6.

Type of Header	**Value Used (Binary)**
Fresh Data Header	00000000 00100001
Retransmitted Interleaved Data Header	00000000 00100010
Retransmitted Fresh Data Header	00000000 00100011
NAK- Interleaved Header	00000000 00100100
NAK- Fresh Header	00000000 00100101

Table 6: Different header values used in the experiment

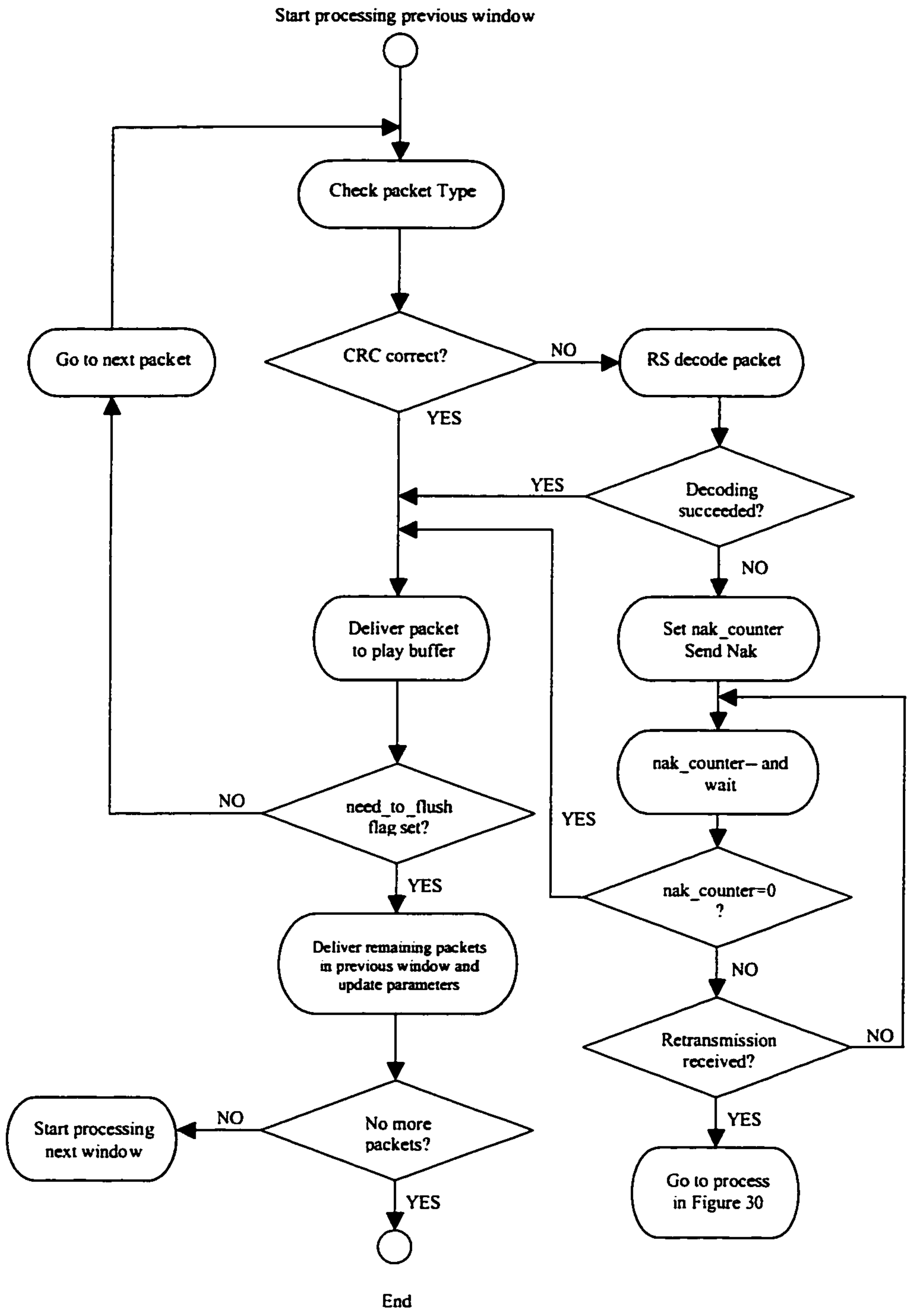


Figure 29: Flow Chart of processing packets of previous window

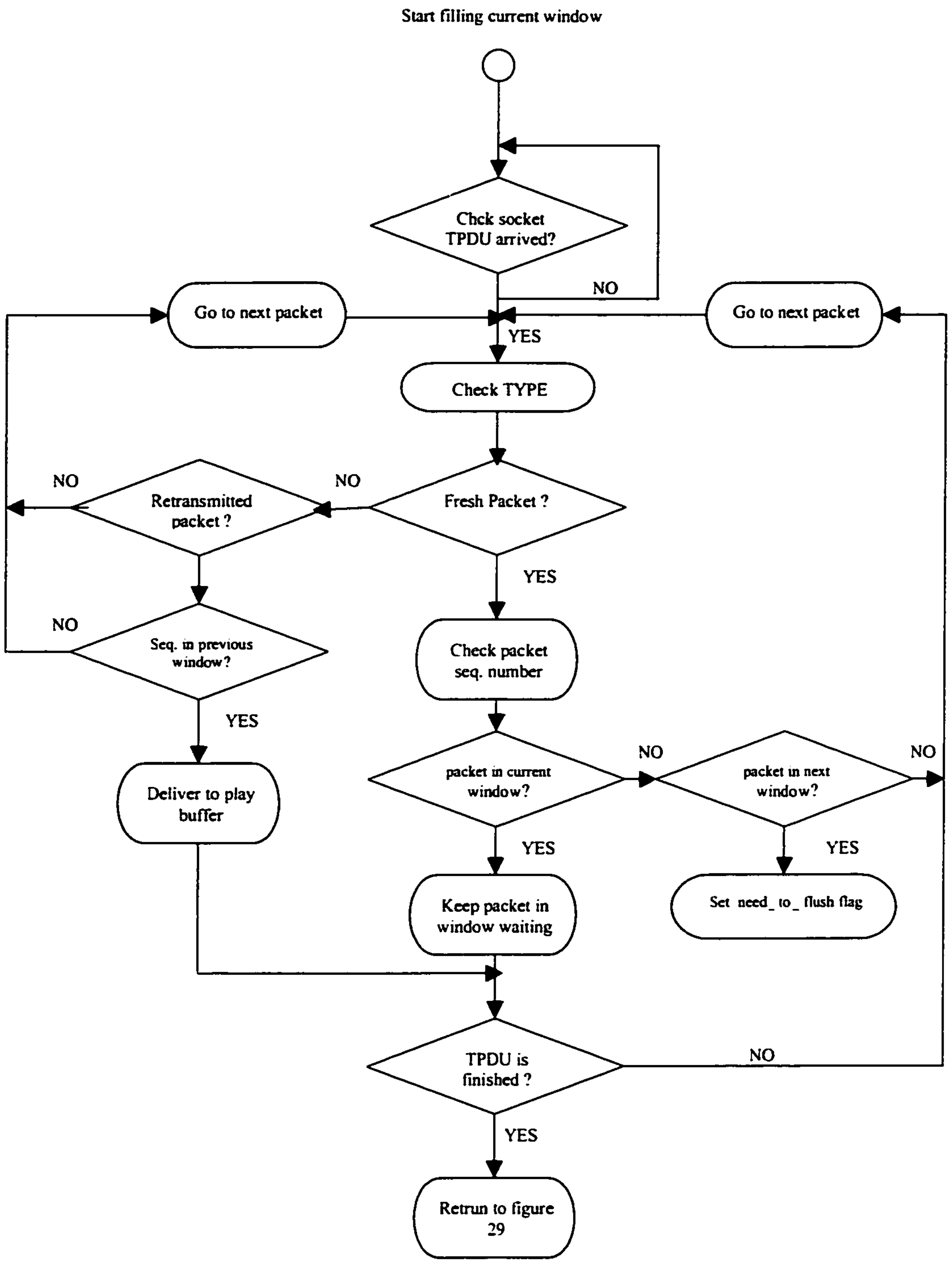


Figure 30: Flow Chart of the process of interrupting/receiving packets in the client

4.4 Implementation of the Proposed Approach

The algorithm described above was realised and translated to C-code software. For experimental purposes, two computers, server and client, was connected peer- to- peer via a 100-base-T Ethernet cable. The server is a 600 MHZ Pentium II machine and the client is a 700 MHZ Pentium II machine. The operating system was chosen to be Red Hat Linux version 6.0. However, the program can be implemented on any version of Red Hat Linux and with some modifications (especially in the TCP/IP Socket programming part), it can be implemented in Windows environment. The communication protocol was chosen to be UDP because it is faster and does not introduce as much overhead as TCP does. Many MPEG players were used to evaluate the quality of the video sequences received; as GTV and MTV players (for Linux environment) and Windows Media Player, VMPEG, and Xing MPEG player (for Windows environment). GTV does not use a method of filtering or error concealment, while other players apply some kind of error concealment (such as repeating the last correct frame to replace any defected frame). The pseudo C-code for the client and server processes is given in the Appendix.

4.5 Experimental Results and Discussion

Many experiments in the labs of ETS were conducted to see the effects of several factors, such as transmission rate, packet loss rate, and the percentages of the 3 types of frames in the two sequences. For the layered Hybrid ARQ/FEC approach, the values for k_I, k_P, and k_B, were set to 191, 207, and 223 respectively, as mentioned before. The *pdf (probability density function)* of packet loss was chosen to be uniformly random, since it is the worst type of loss in networks. Moreover, it is more difficult for MPEG players to conceal random errors than to conceal burst errors [16]. The packet loss was changed between 0-15 percent. In addition to the effect of packet loss, the effect of transmission rate is also noticed. It has to be mentioned that no flow control mechanisms are applied here, and this issue should be searched in future research.

When the two sequences are encoded using the Layered Hybrid ARQ/FEC approach, their sizes increase due to the introduction of parity bytes, encoding header bytes, and interleaving header bytes. Encoding Miss America sequence led to almost 32 % overhead, and encoding Flower Garden sequence led to 27% overhead.

The results of the experiments are shown in the Figures 31-48 and the discussion of these results will be explained in details. It should be mentioned that each point of the results was got by averaging the results of 3 different runs of the client/server program.

4.5.1 FEC Role in Error Recovery

Figures 31 and 32 show the percentage of correctly decoded B packets (the packets where data part belongs to a B Frame) for Miss America and Flower sequences, respectively. Two factors affect the percentage of correctly decoded B packets; transmission rate and packet loss percentage.

It can be noticed that transmission rate plays a key role in determining the amount of correctly decoded packets. For instance, Figure 31 shows that, for Miss America sequence, when the network packet loss rate is 3 %, the transmission rate can be increased up to 1.5 Mbps and still the RS decoder would be able to correct all the

B packets with errors/erasures. However, when the transmission rate goes beyond 1.5 Mbps, the decoder would not be able to decode all the B packets (before the *next window* is detected), and the non-decoded packets will be delivered as they are to the playing buffer. Hence, it is natural that the number of correctly decoded B packets will start decreasing. For the same packet loss rate, Figure 32 shows that, for Flower Garden sequence, the *maximum transmission rate* (the maximum transmission rate that can be achieved, for a specific packet loss rate, without delivering any packet with possible errors to the playing buffer) would be around 1.8 Mbps. The maximum transmission rates for some packet loss percentages are shown in Table 7. The same analysis can be done about the P and I packets in Figures 33-34 and Figures 35-36, respectively.

Packet Loss %	Maximum Transmission Rates for Miss America (Mbps)	Maximum Transmission Rates for Flower Garden (Mbps)
1	1.77	1.822
3	1.5	1.79
5	1.4	1.5
8	1.3	1.26
10	1.2	0.970

Table 7: Some maximum transmission rates for the two MPEG sequences

Packet loss percentage also has limiting effects on the performance of the real implementation. When the average percentage of packet loss exceeds the limit of 12.5% (which corresponds to the 32 erasures/16 errors that can be recovered by RS decoder for B packets), not all the B packets can be decoded correctly no matter how low the transmission rate is. This is clearly shown in Figures 31-32. For Miss America sequence, when the packet loss rate is 13 %, the percentage of correctly decoded B packets would decrease, even when the transmission rate is as low as 600 Kbps, from 27% to 14 %. This difference is the amount of B packets that could not be decoded and Fresh NAKs have to be initiated for them. On the other hand, for P and I packets in the Figures 33-34 and 35-36 respectively, a packet loss error of 13 % would not affect the percentage of decoded packet, since P and I packet can tolerate up to 18.75 % and 25 % packet loss, respectively. It should be mentioned that if the lower part of the sequence number of at least one lost interleaved packet is smaller than k_I, *all* the packets in a window must be decoded. That is because of the chosen interleaving depth. If the interleaving depth were kept smaller than 256, not all packets in a window need to be decoded if the sequence number in a window is less than k_I.

4.5.2 NAK/ARQ Role in Error Recovery

When a packet fails to be RS decoded (typically when the packet loss rate exceeds 12.5%), the client will send a *Fresh NAK* packet to the server and hence this packet is not considered as a correctly decoded packet. As the packet loss percentage increases, more packets are destroyed and the number of packet that should be retransmitted increases. This can be shown in Figures 41 and 42. When the packet loss rate is 13 %, for Miss America sequence in Figure 41, around 10% percent of all packets are NAKed. When the packet loss rate is increased to 15%, almost 15 % of the packets should be recovered by ARQ. If the packet lost percentage is increased, the percentage of NAKs sent will be rising until it saturates when *all* B packets are asked to be retransmitted. The same analysis can be made for Figure 42 for Flower Garden sequence. However, it can be noticed that the percentage of NAKs sent are less than that of the first sequence; i.e. for 15 % packet loss, only 6% of packets could not be decoded and have to be retransmitted. That is mainly because of the different ratios of B bytes in the original sequence; while 31.60% of the bytes of Miss America are B bytes, only 8.91% of the bytes in Flower Garden are B bytes. This difference in ratios will normally make Miss America more sensitive to high packet loss rates.

The two following figures (43 and 44) show the percentages of retransmitted packets to the client. After a NAK is transmitted, the client waits for around 30 ms for retransmission, and if no retransmission is detected, the packet is delivered as is. During the experiments, it was not possible to retransmit all the packets requested to the receiver because of many reasons such as programming limitations, transmission rate effects, number of packets requested, and rate of sending NAKs. In order to recover sustainable amount of the requested packets, the transmission rate should be decreased substantially. In Miss America, at 15 % packet loss rate, almost 4 % of all the packets are retransmission packets. This means that only 26.6 % of the NAK requests sent were recovered using ARQ. Moreover, this ratio could not be achieved unless the transmission rate was decreased to 500 Kbps. On the other hand, in Flower Garden at the same loss rate, almost all the requested packets were retransmitted. That is because of the difference of the number of B packet that could not be RS decoded.

Figures 39 and 40 show how the percentage of packets dropped or/and delivered with possible errors in the two MPEG sequences. It should be noted that these figures do not show the residual error. They show simply how many *packets* were not delivered correctly and error-free to the playing buffer. As it can be noticed, this number increases as the transmission rate exceeds the maximum transmission rate and as the packet loss rate exceeds the threshold of 12.5%.

Figures 45 - 46 give the residual error- *which is the total number of bytes delivered in error divided by the total number of bytes sent* and Figures 47 - 48 show the total throughput, *which it the total number of correct bytes delivered divided by to the total number of bytes sent*, for the two MPEG sequences. The subjective quality of the resultant video sequence will be discussed in the next chapter.

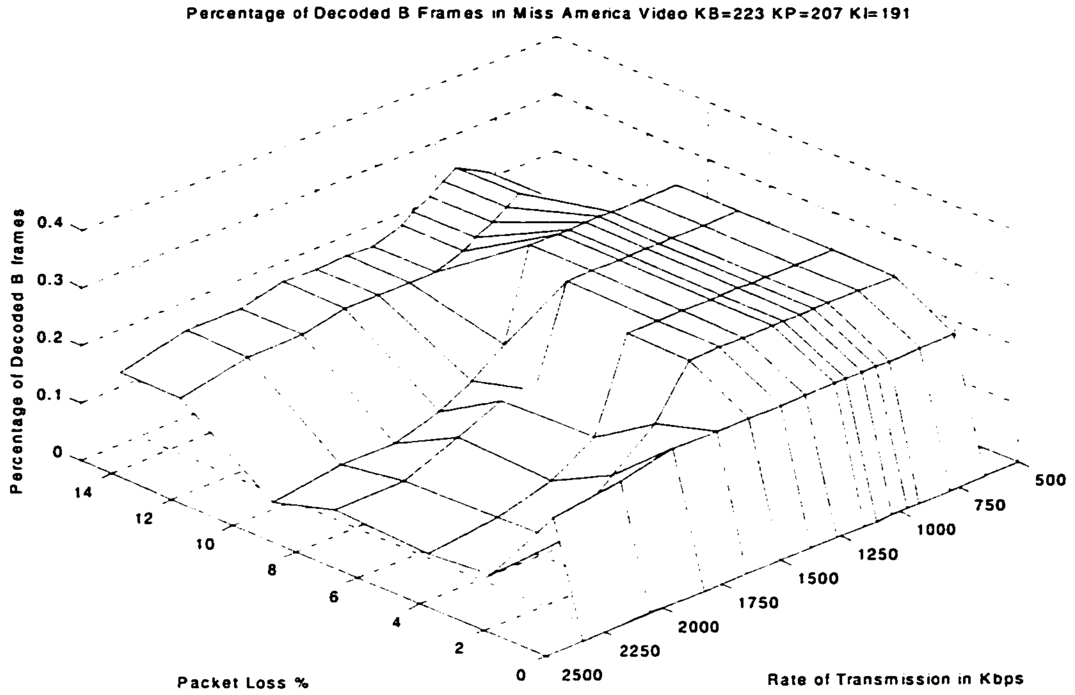


Figure 31: Percentage of Correctly decoded B packets in Miss America using Layered Approach

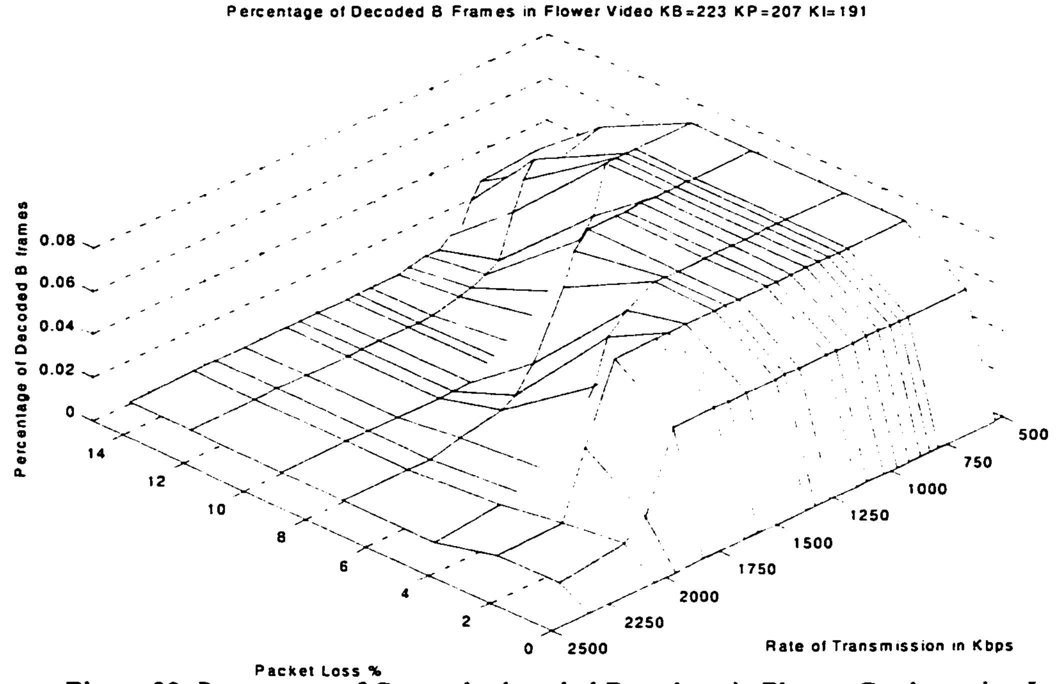


Figure 32: Percentage of Correctly decoded B packets in Flower Garden using Layered Approach

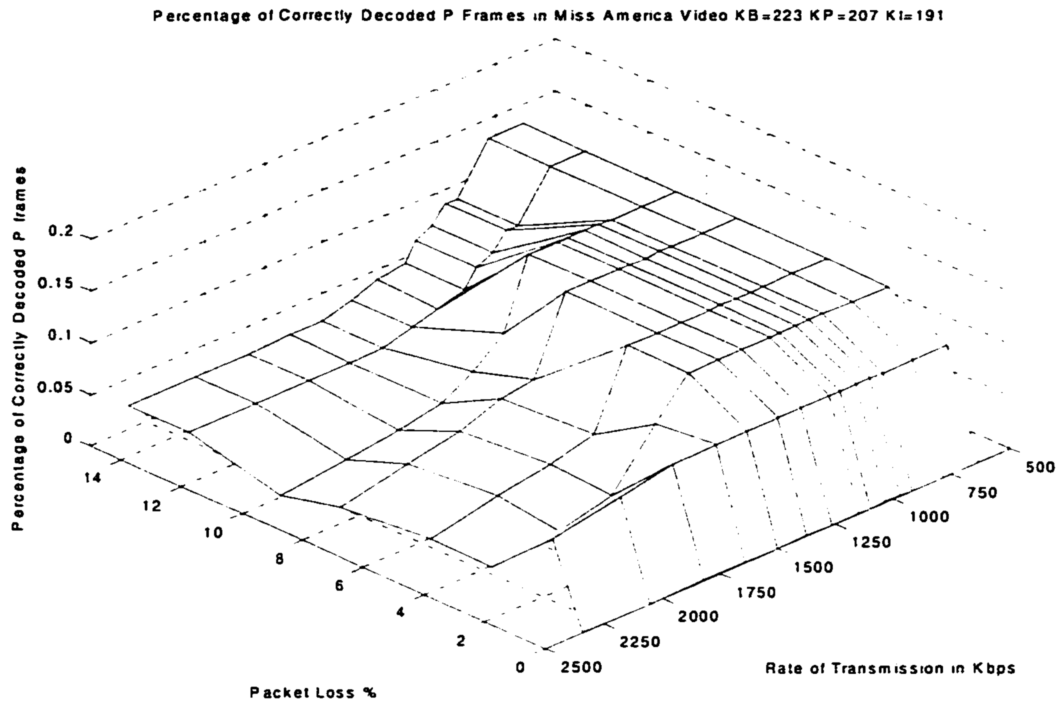


Figure 33: Percentage of Correctly decoded P packets in Miss America using Layered Approach

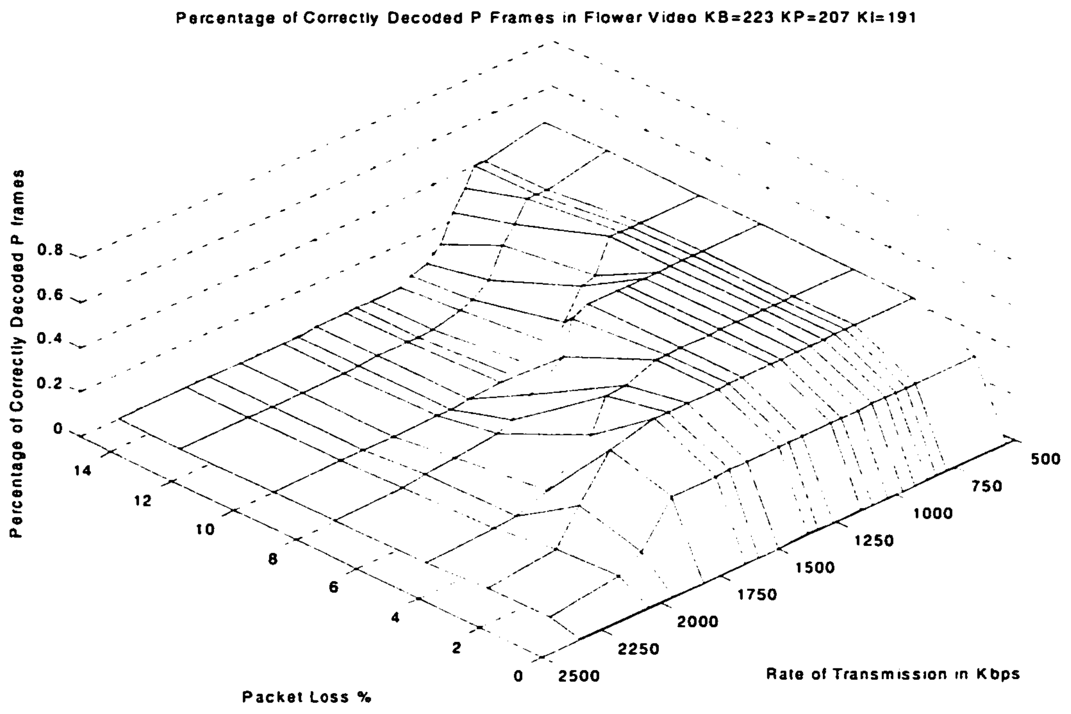


Figure 34: Percentage of Correctly decoded P packets in Flower Garden using Layered Approach

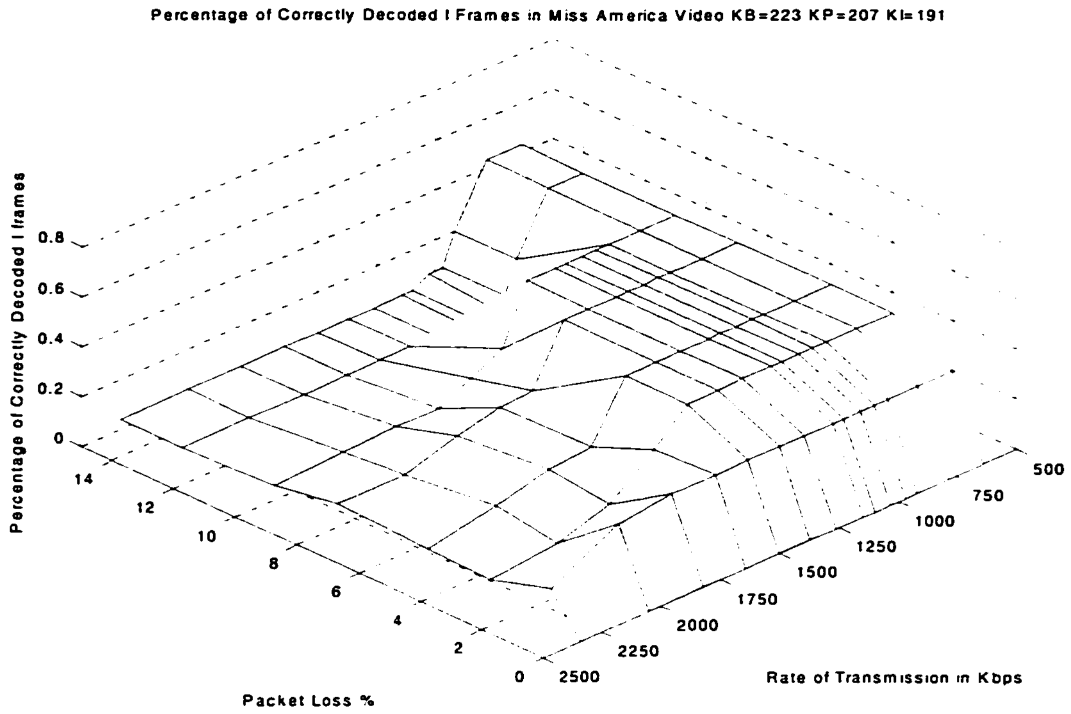


Figure 35: Percentage of Correctly decoded I packets in Miss America using Layered Approach

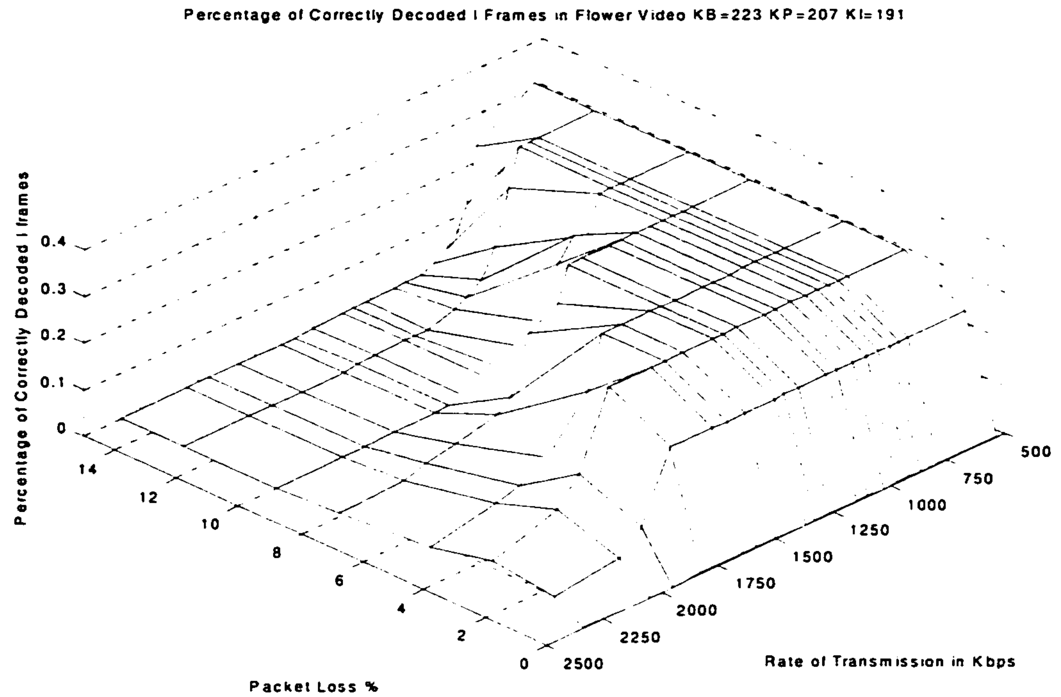


Figure 36: Percentage of Correctly decoded I packets in Flower Garden using Layered Approach

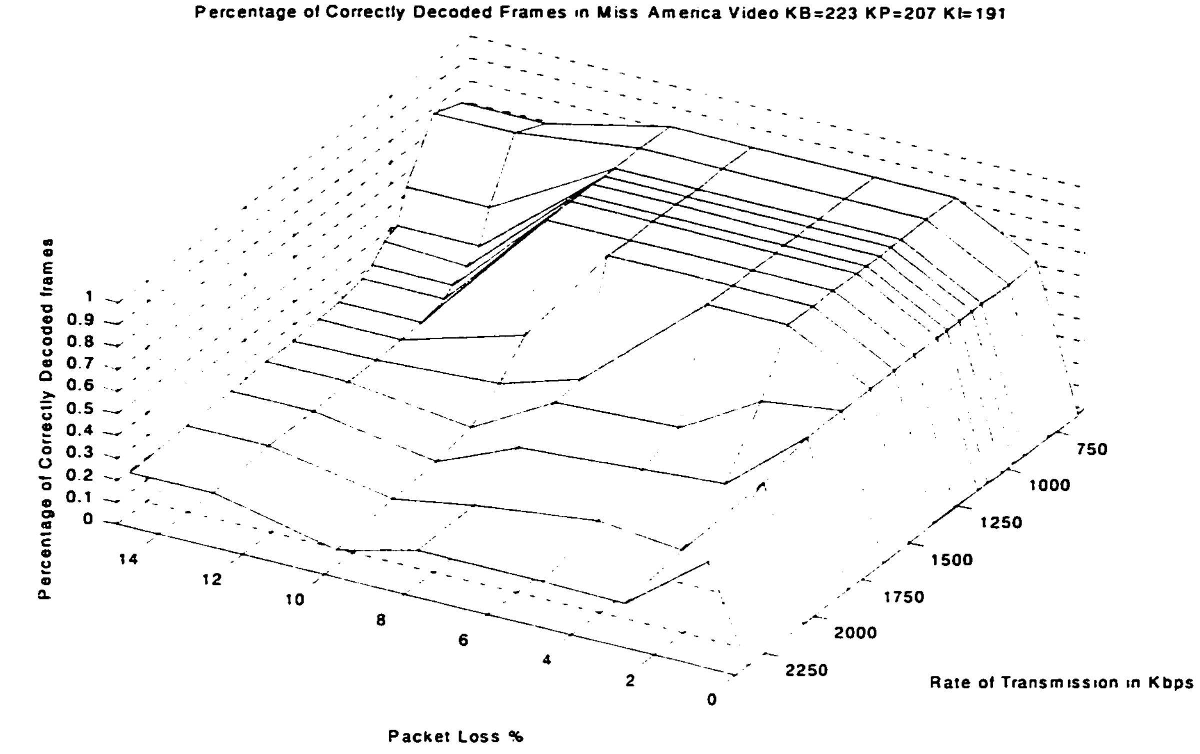


Figure 37: Percentage of Correctly decoded packets in Miss America using Layered Approach

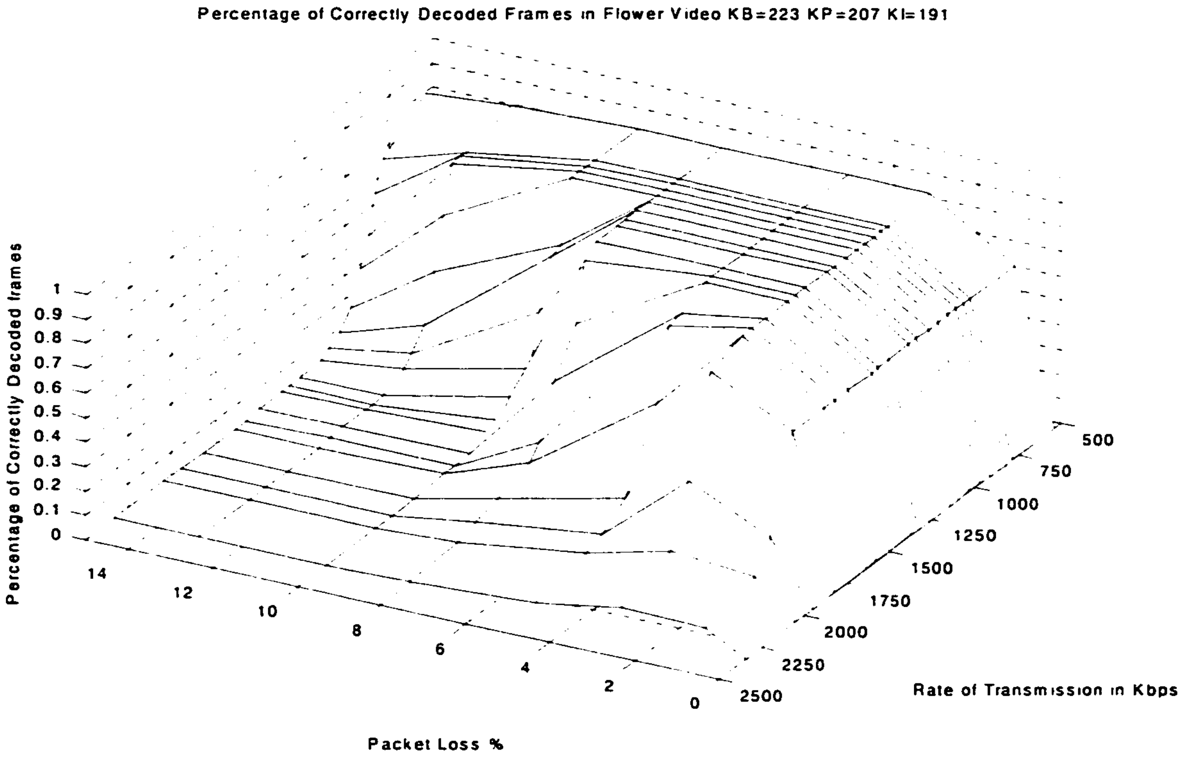


Figure 38: Percentage of Correctly decoded packets in Flower Garden using Layered Approach

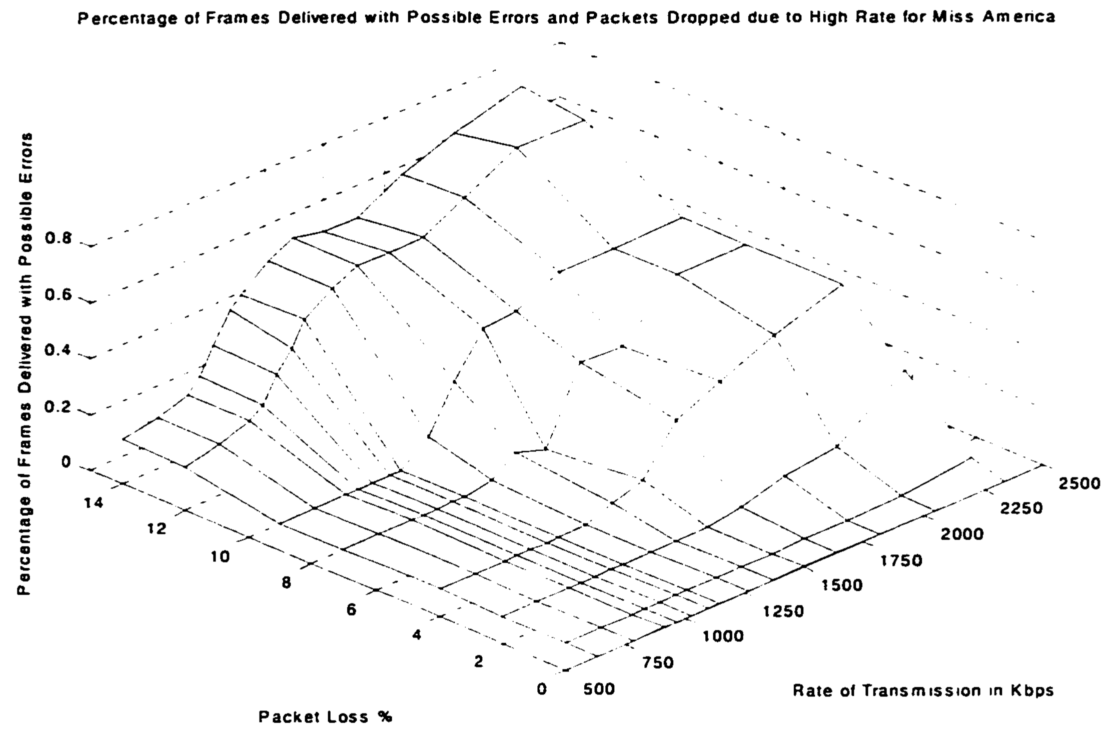


Figure 39: Percentage of packets dropped or delivered with possible errors in Miss America using Layered Approach

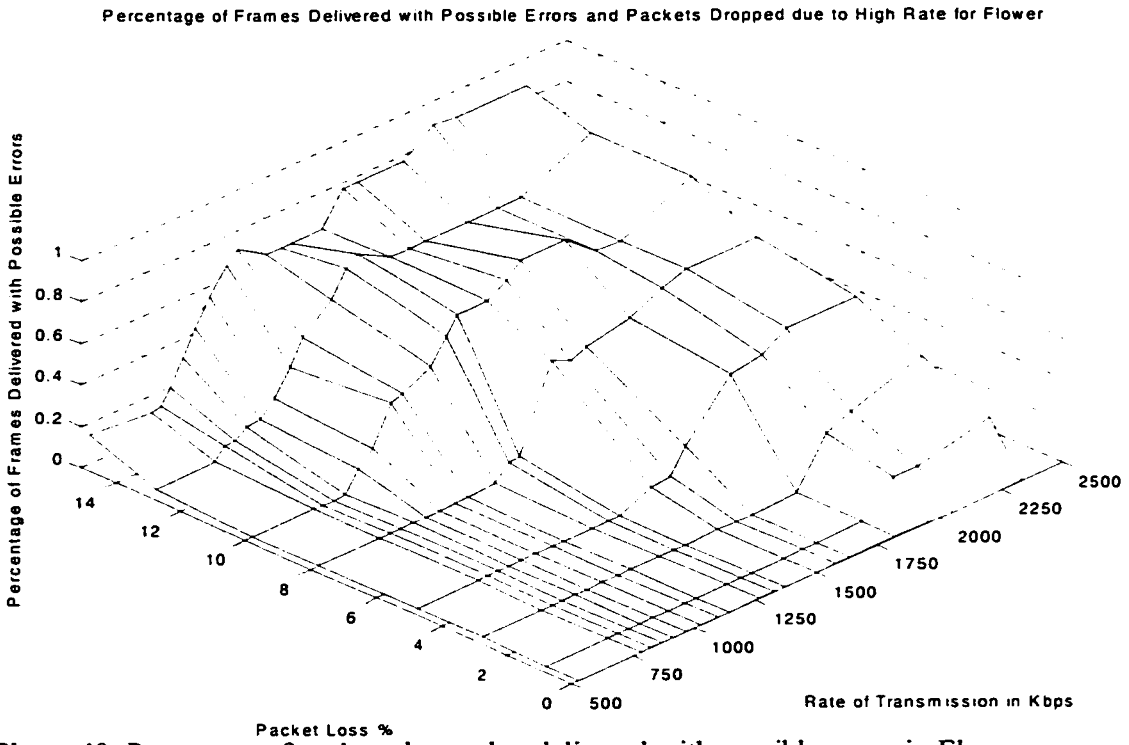


Figure 40: Percentage of packets dropped or delivered with possible errors in Flower Garden using Layered Approach

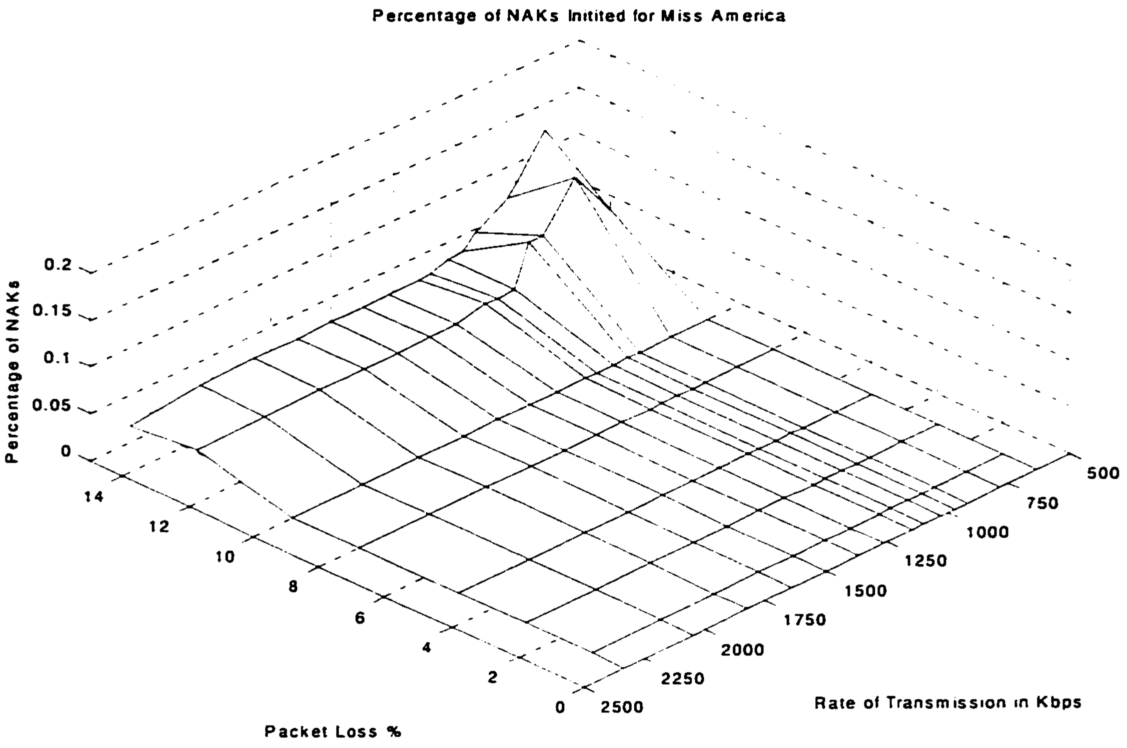


Figure 41: Percentage of NAKs sent in Miss America using Layered Approach

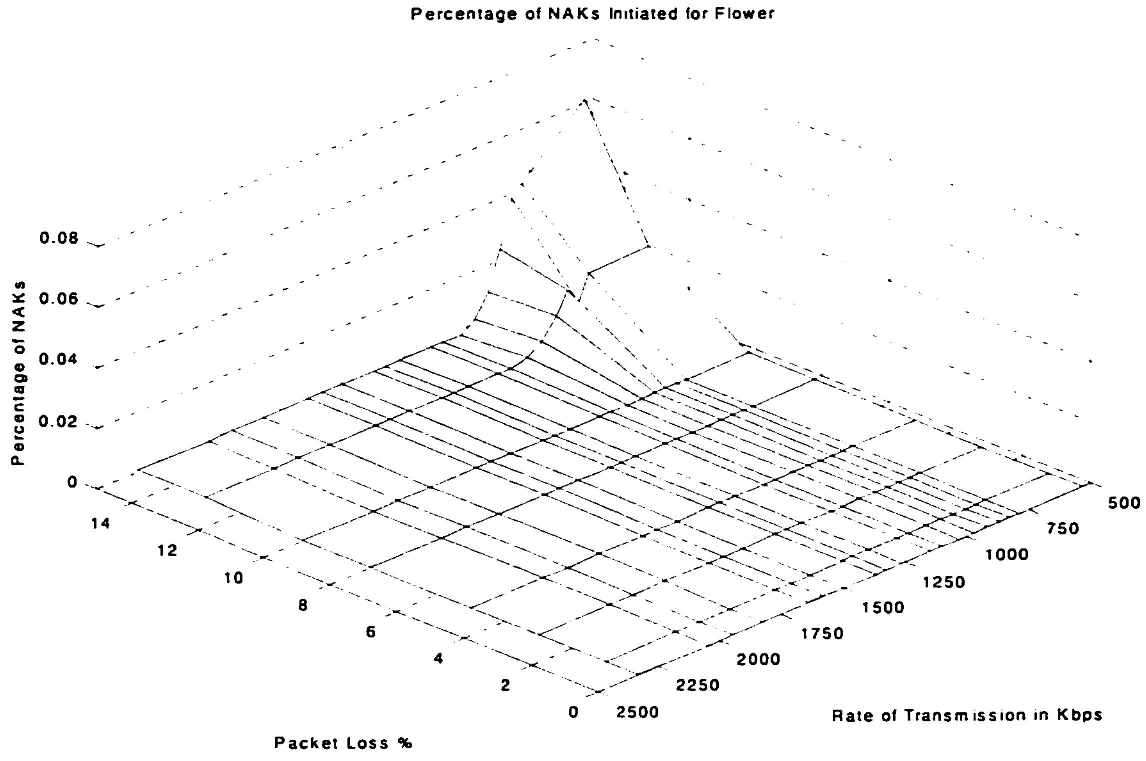


Figure 42: Percentage of NAKs sent in Flower Garden using Layered Approach

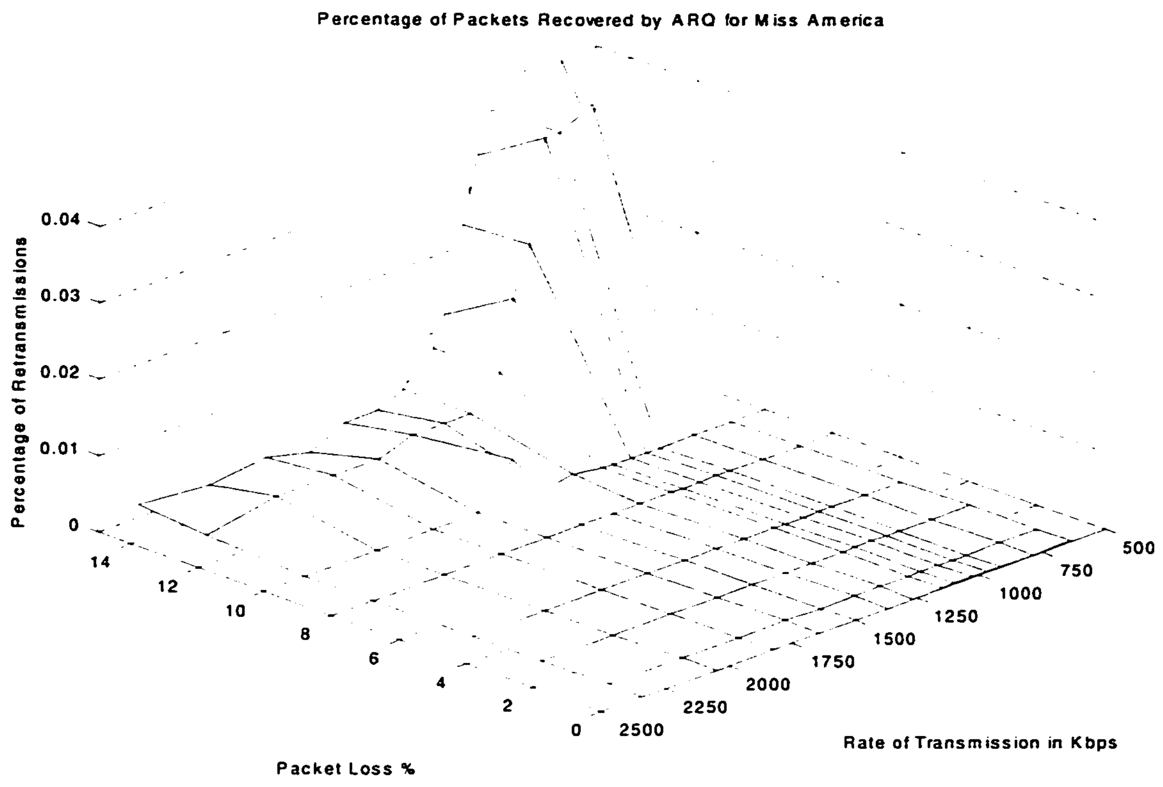


Figure 43: Percentage of retransmissions in Miss America using Layered Approach

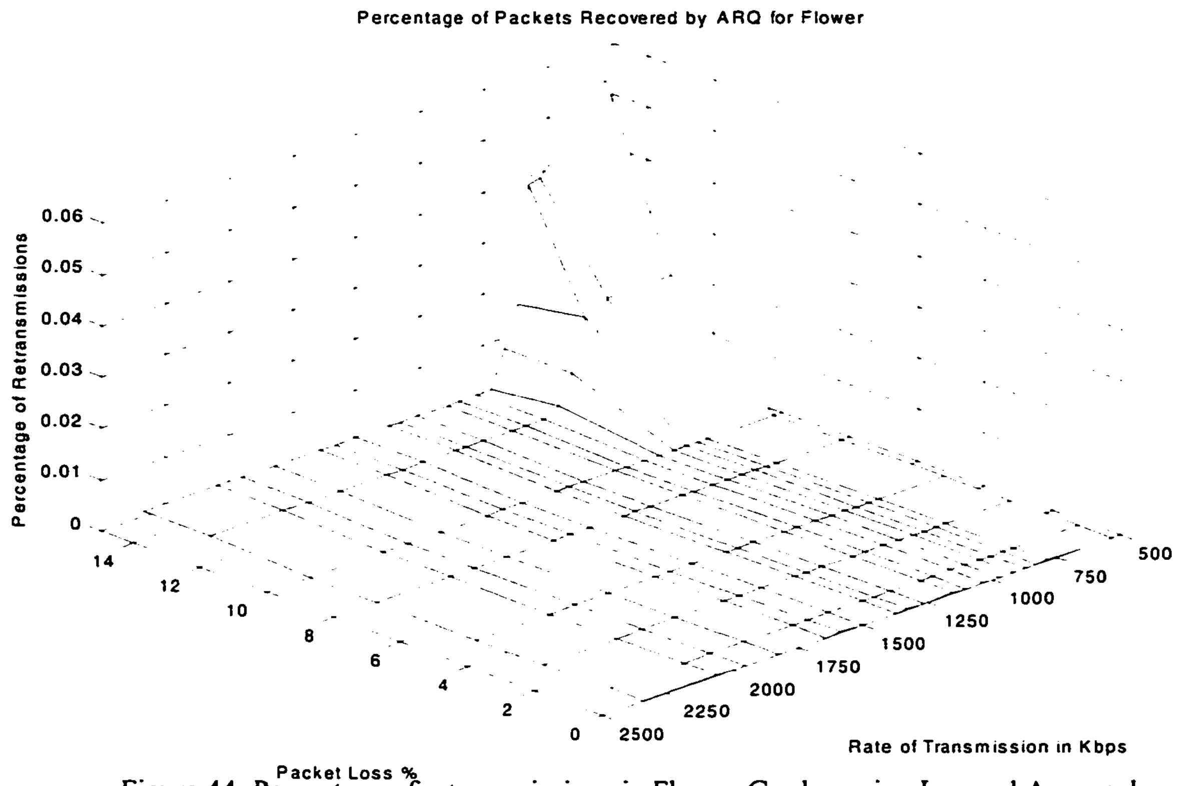


Figure 44: Percentage of retransmissions in Flower Garden using Layered Approach

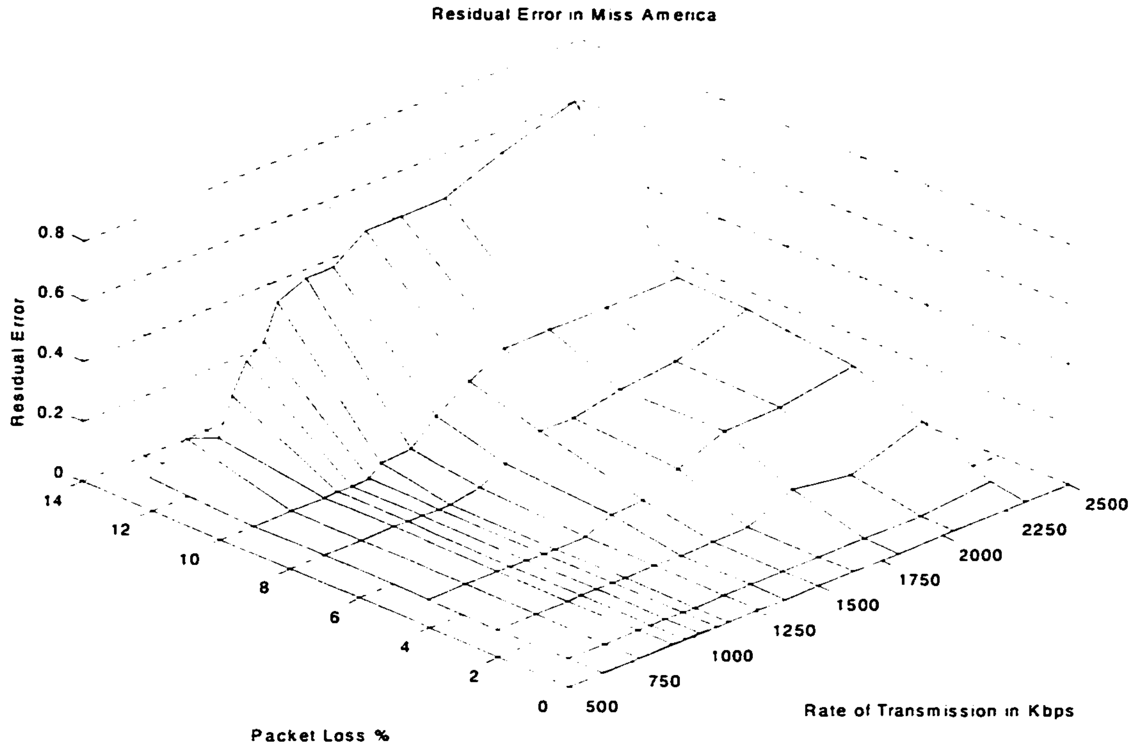


Figure 45: Residual error in Miss America using Layered Approach

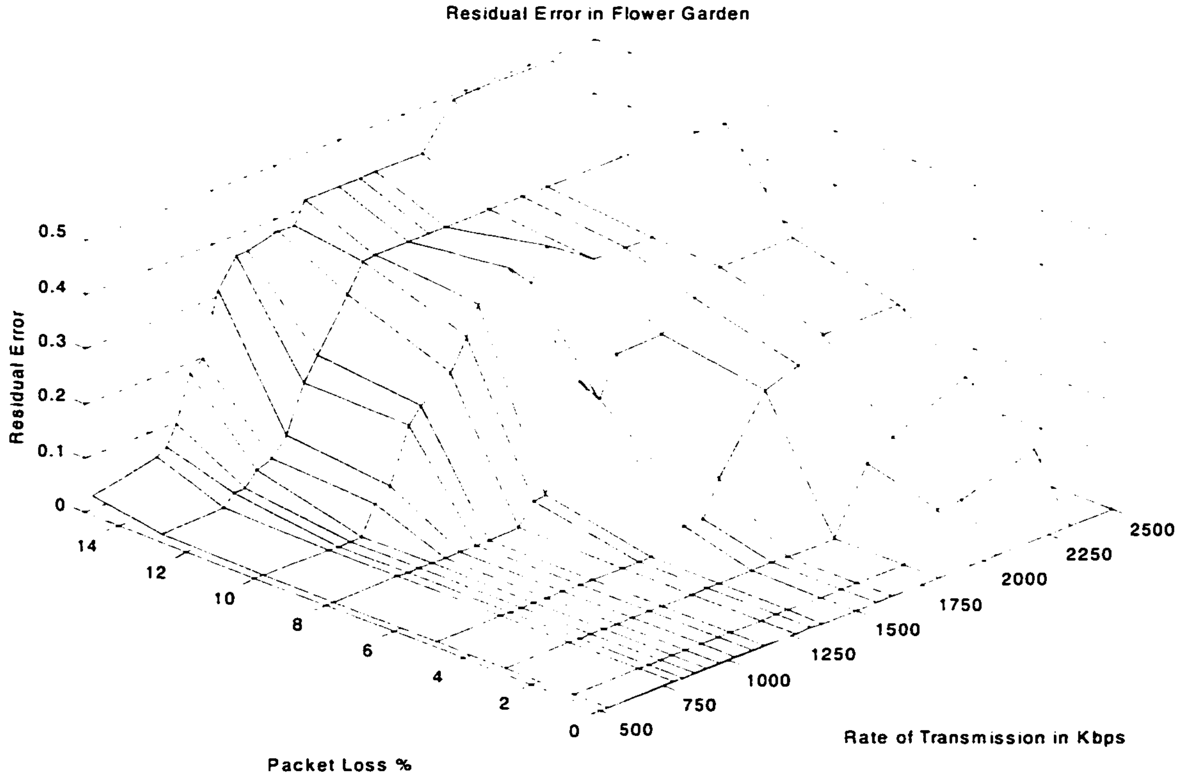


Figure 46: Residual error in Flower Garden using Layered Approach

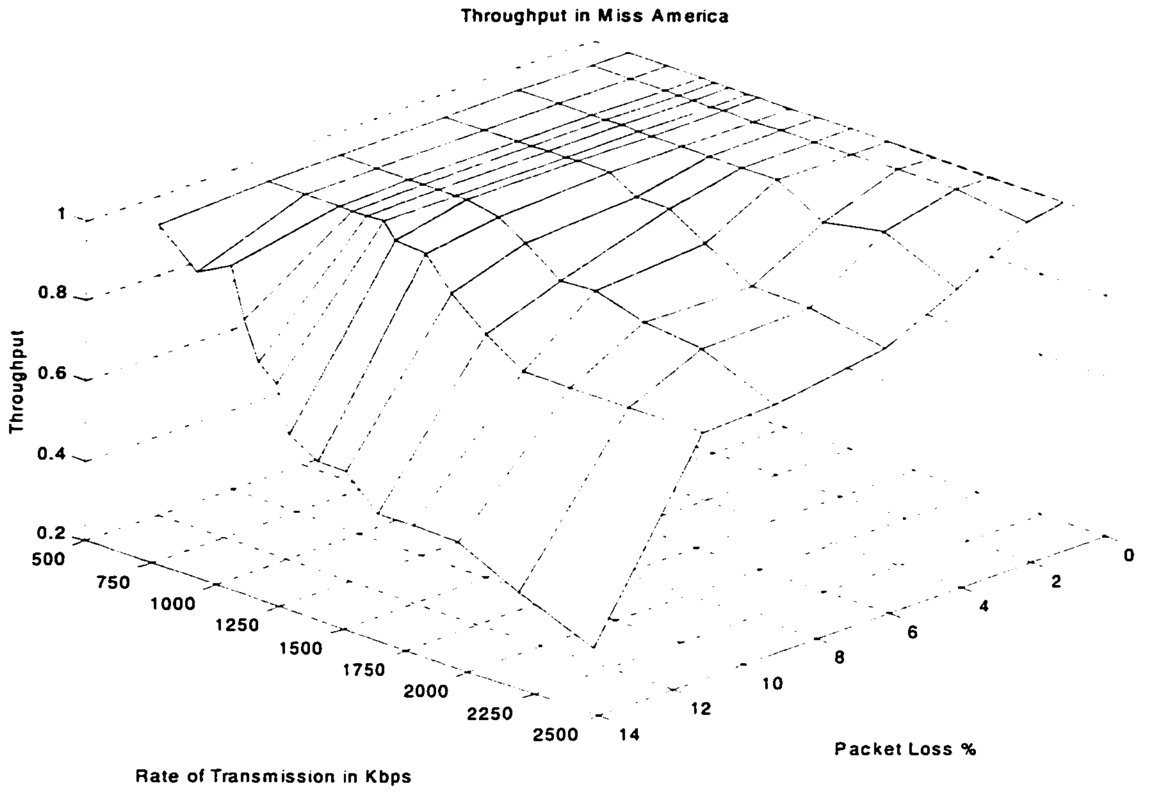


Figure 47: Throughput in Miss America using Layered Approach

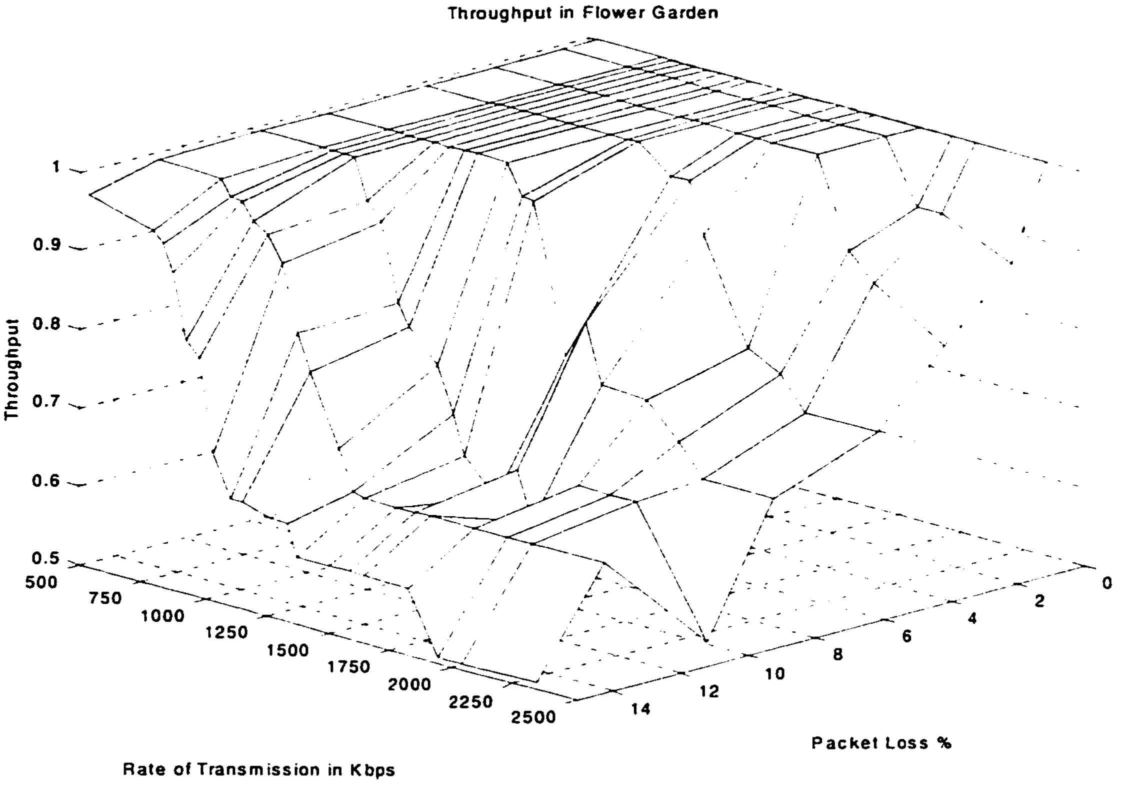


Figure 48: Throughput in Flower Garden using Layered Approach

CHAPTER 5

Subjective Evaluation of the Proposed Approach

This part deals with the evaluation of the results previously achieved in chapter 4. In this chapter, the benefits of applying the Layered Hybrid ARQ/FEC approach will be discussed and compared with other classical methods. Also, the limitations and the drawbacks of this approach will be discussed and future work would be highlighted.

5.1 Quality of Picture when Implementing Layered Hybrid ARQ/FEC

The results shown in Figures 31-48, when implementing Layered Hybrid ARQ/FEC method, would be reflected on the quality of the reconstructed MPEG sequence at the receiver. Subjective evaluation for the quality of images shows the following:

- As long as the residual error is less than 10%, the resulting sequence would not endure any noticeable or long lasting errors. This situation often occurs when the transmission rate exceeds slightly the maximum transmission rate. The quality of image would range from very good to perfect
- As long as the residual error is between 10% and 15%, the resulting sequence would endure noticeable, but not dramatic, errors and more slices would be affected. The quality of image would be acceptable and the video would be comprehensible.

- When the residual error is ranging between 15% and 20%, more parts of the image would be missing and some MPEG players (like Windows Media Player) would start considering the destroyed frames as missing and start replacing it by the previous correct frame. The video sequence would be tolerable and it would be annoying to some extent.
- If the residual error exceeds the limits of 20%, it was noticed that the quality of image would deteriorate and the quality of image would fluctuate between annoying and not acceptable. This case generally occurs when the packet loss rate exceeds the maximum threshold allowable, which is in this case 12.5%, and when the transmission rate is relatively high. Some screen shots of the resulting videos are shown in Figure 49.

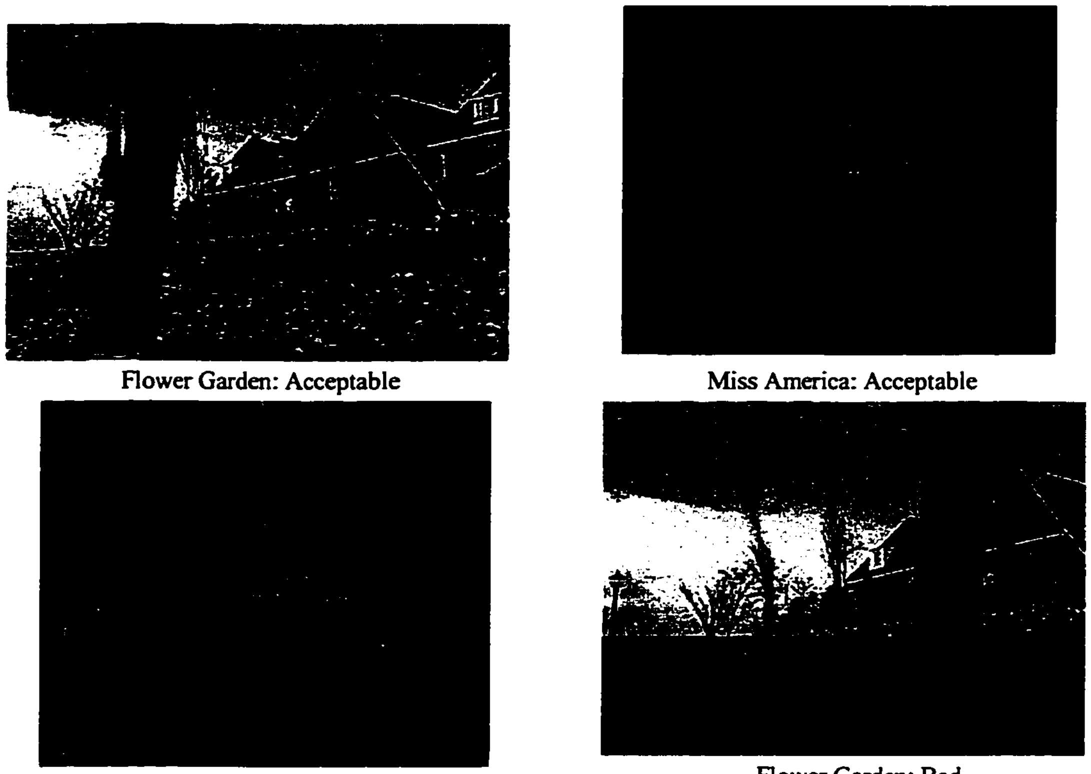


Figure 49: Some screen shots with different qualities

It should be noted that residual error was chosen, in this thesis, as a criteria for measuring the image QoS. This method is one of many others that are used to determine the purity of an MPEG image, just like the PSNR and MLD [23]. For example, Peak Signal to Noise Ratio (PSNR) is a very common way to evaluate the quality of images. PSNR is applied on uncompressed sequences by calculating the MSE (Mean Square Error) between the original sequence and decoded sequence [11].

$$PSNR = -10\log_{10}\left(\frac{MSE}{2^n - 1}\right)^2$$

where n is the number of bits used to represent each sample in a digitised frame. The PSNR measurement should be investigated in the future, the thing that can give a clearer idea about the QoS.

5.2 Comparing with Pure ARQ Approach

Despite of the advantages of using ARQ in low error/loss rate mediums, ARQ becomes in adequate where the packet loss rate rises significantly [19] and the throughput decreases significantly if the transmitter is kept busy with retransmission [20]. Generally speaking, in a pure ARQ environment the percentage number of packets needed to be retransmitted will have a linear relation (with a slope =1) with the percentage of packet loss [21]. On the other hand, if the Layered Hybrid ARQ/FEC were implemented, the percentage of retransmission *would not* be linearly related to the packet loss rate. Instead, it would depend on either of two factors:

1. The number of parity symbols (bytes in this thesis) assigned for each packet
2. The percentage of bytes that would need retransmissions according to priority distribution specified for each frame type (mostly B bytes in this thesis)

Figure 50 plots the *theoretical* percentage NAK requests in a pure ARQ environment versus the percentage of NAK requests in a Layered Hybrid ARQ/FEC, which was achieved during the tests in ETS. It can be shown in Figure 50, that the number of packets that need to be retransmitted in the layered approach is almost zero when the packet loss rate is below 12.5 %. That is of course due to the RS decoding capability of the implemented approach. When the packet loss rate goes beyond the 12.5%, the number NAKs rises significantly to reach 15% for 15% packet loss rate in Miss America, and 6% for 16% packet loss rate in Flower Garden. In either case, the Layered Hybrid ARQ/FEC method behaves more efficiently than pure ARQ method, and it would downsize hugely the traffic on the network, especially at lower packet

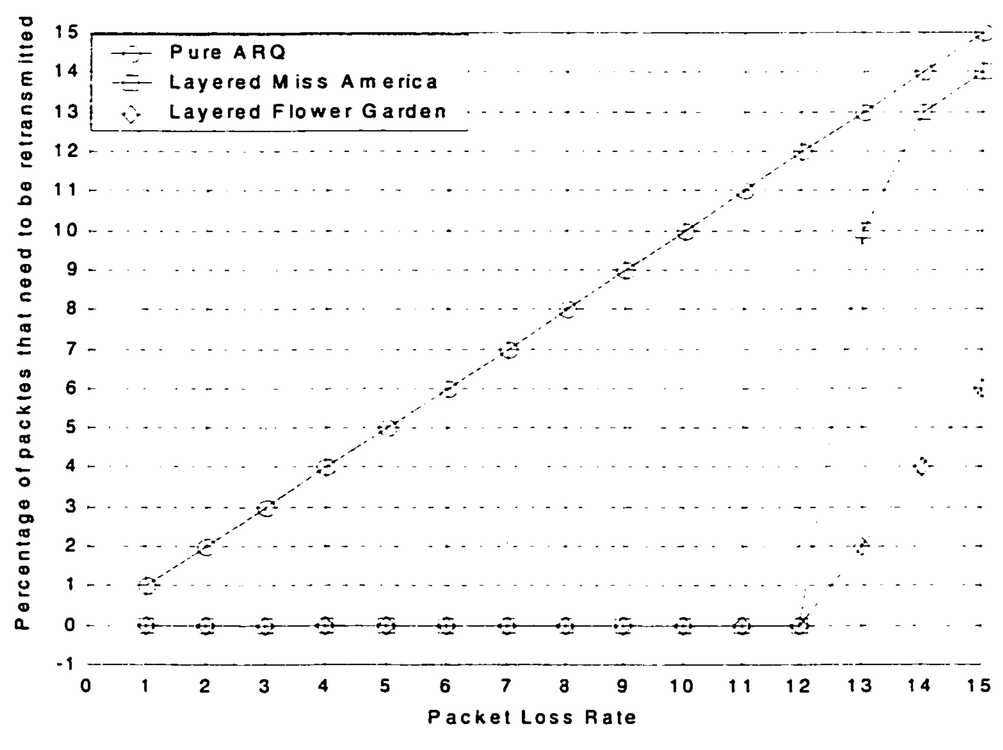


Figure 50: Comparison in number of retransmission needed between pure ARQ (theoretically) and Layered Hybrid approach

5.3 Comparing with Non-Layered Hybrid ARQ/FEC

The benefits of Layered Hybrid ARQ/FEC were further investigated and a comparison with Non-Layered Hybrid ARQ/FEC was studied in this thesis. Actual testing (in ETS labs) for the non-layered approach was performed on Flower Garden MPEG sequence with k=223 for all packets, no matter to what frame a packet belongs (I, P, or B Frames). It would not cause noticeable difference if the testing were conducted on Miss America sequence, since all packets would also have the same priority level. The results of the experiments conducted in ETS labs are shown in the Figures 51-56.

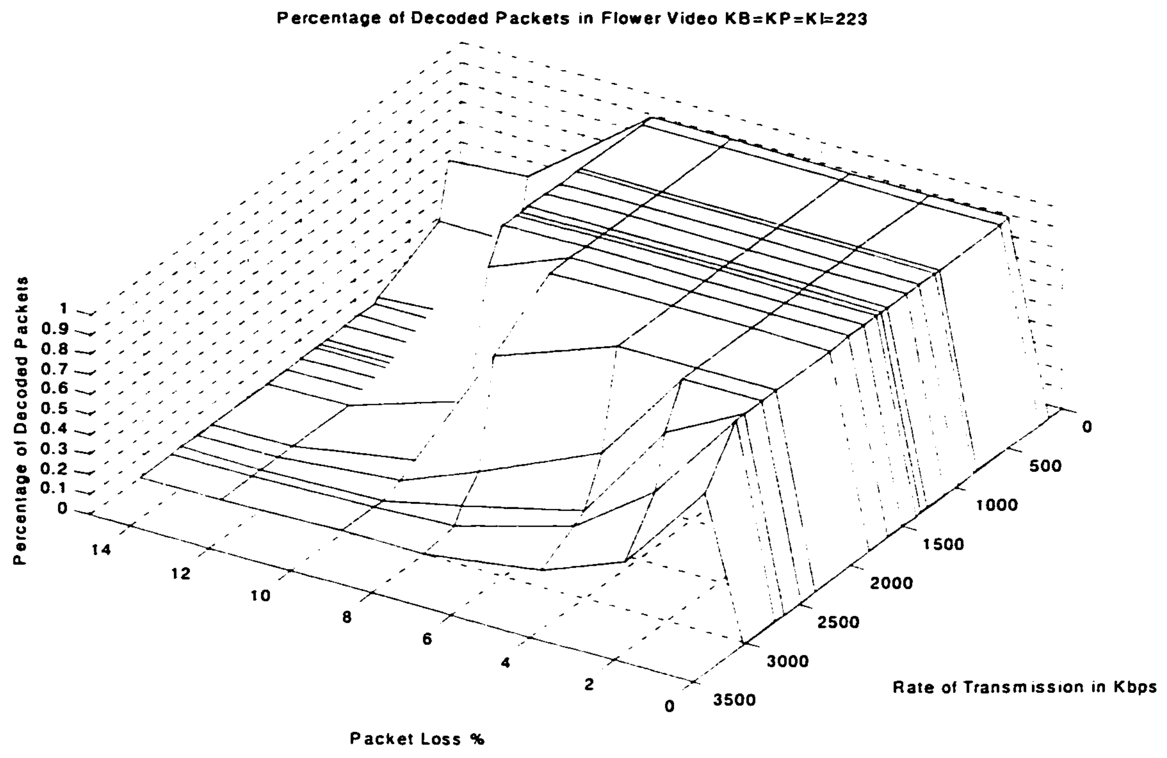


Figure 51: Percentage of Correctly decoded packets in Flower Garden using non-layered Approach

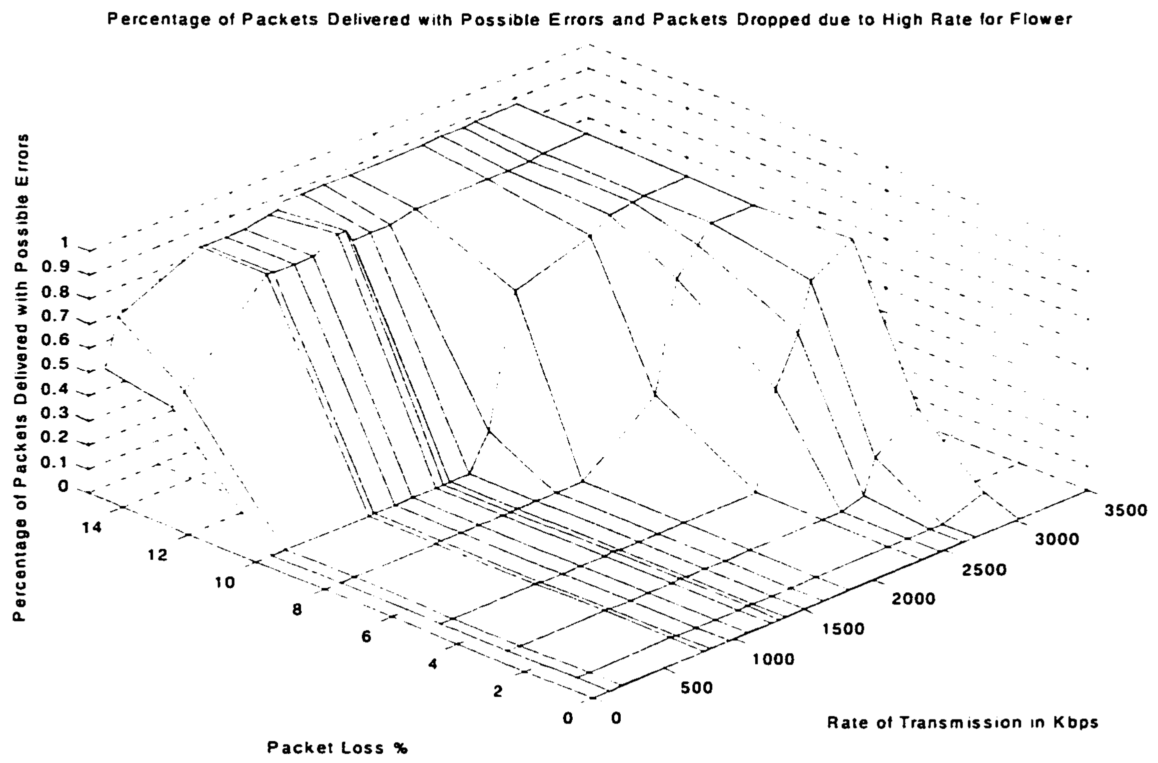


Figure 52: Percentage of packets dropped/delivered in possible errors in Flower Garden using non-layered Approach

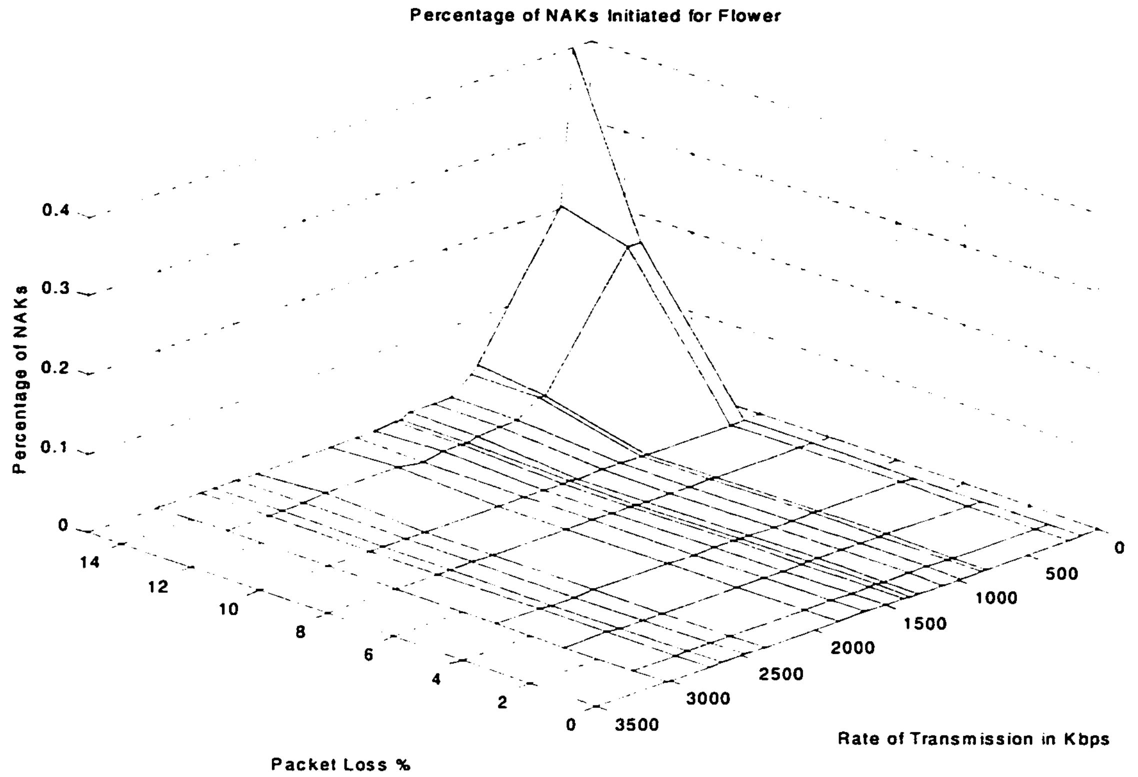


Figure 53: Percentage of NAKs initiated in Flower Garden using non-layered Approach

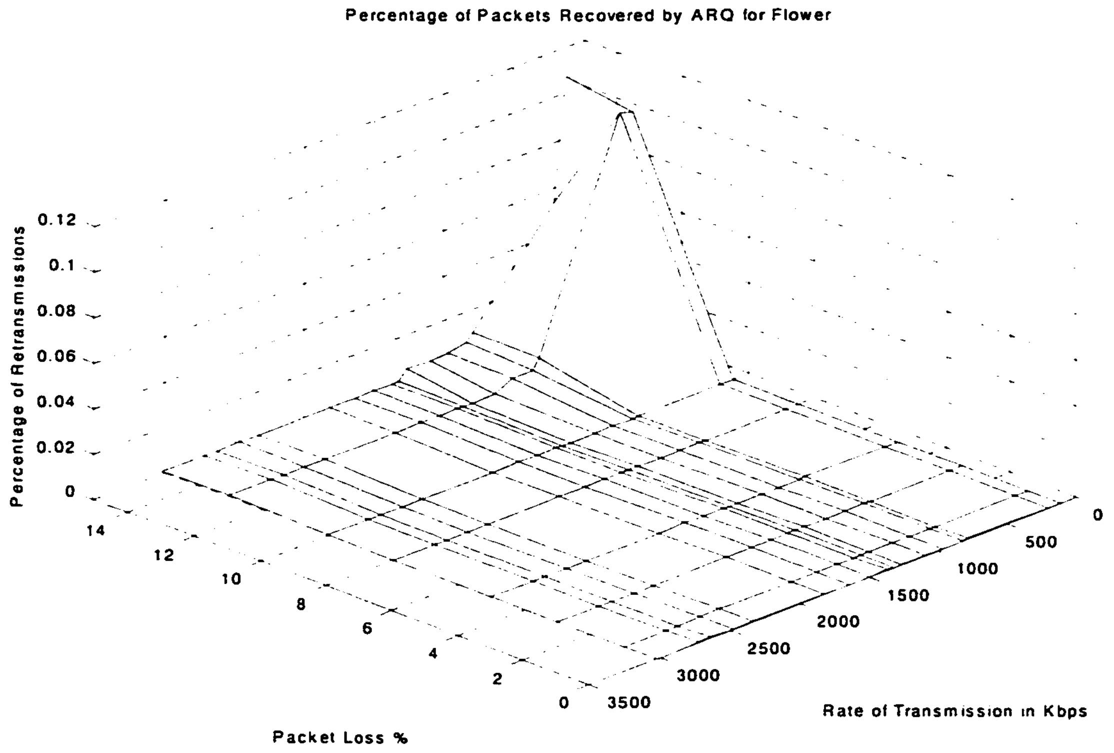


Figure 54: Percentage of packets recovered by ARQ in Flower Garden using non-layered Approach

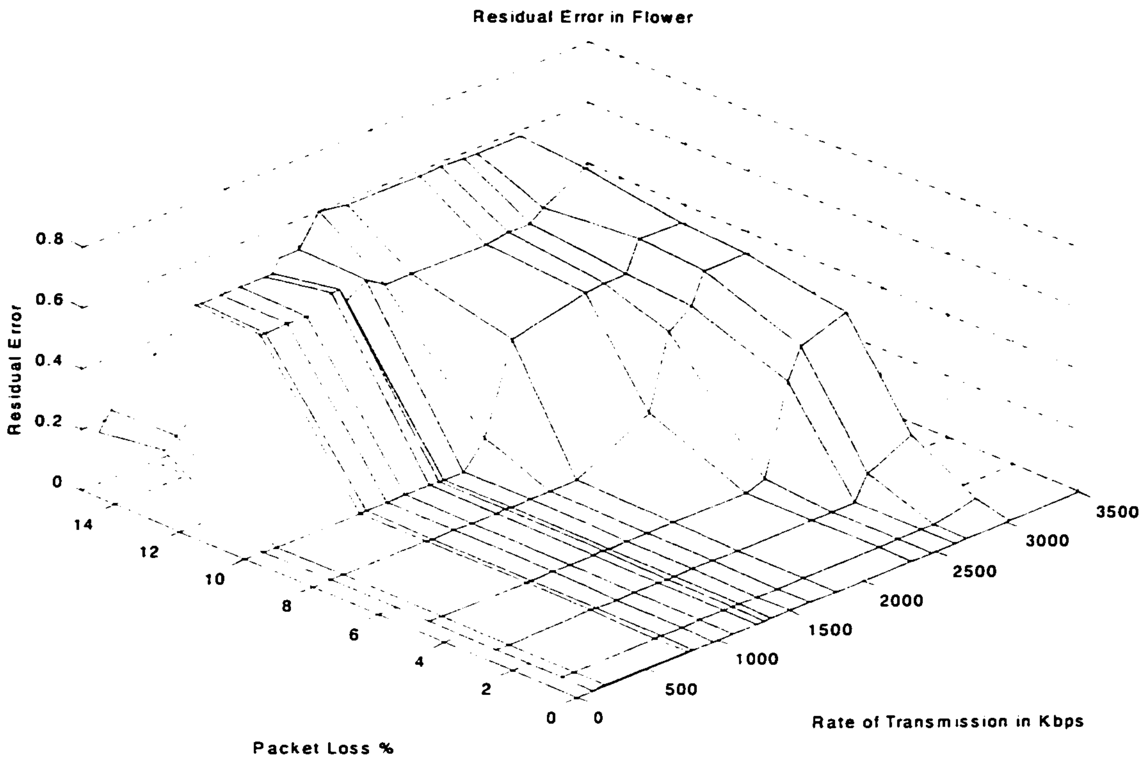


Figure 55: Residual error in Flower Garden using non-Layered Approach

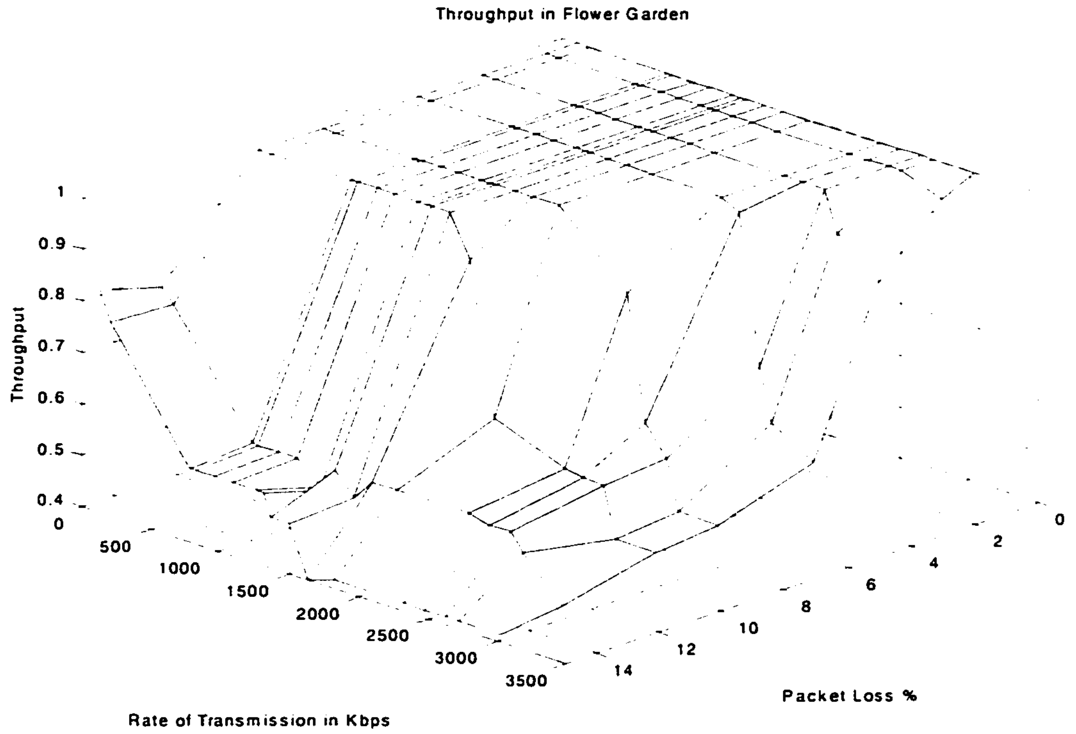


Figure 56: Throughput in Flower Garden using non-Layered Approach

1. The maximum transmission rate for non-layered approach is *higher* than that of the layered approach proposed. This is due to the fact that the RS decoder can decode (223,255) packets with faster rates than that of (191,255) or (207,255) packets. Comparison in maximum transmission rates are shown in Table 8

Packet Loss %	Maximum Transmission Rates for non-Layered Flower Garden (Mbps)	Maximum Transmission Rates for Layered Flower Garden (Mbps)	Percentage of rate increment using non layered approach
1	2.8	1.822	0.53
3	2.5	1.79	0.39
5	2.35	1.5	0.56
8	1.8	1.26	0.42
10	1.5	0.970	0.54

Table 8: Comparison between maximum transmission rates for the Flower Garden sequence using layered and non-layered approaches

2. Despite the approximate 50% gain in maximum transmission rate, the non-layered has many disadvantages. It can be noticed that the number of packets dropped/delivered in error rises rapidly and more steeply than the layered approach as soon as the transmission rate exceeds the maximum. This would affect rapidly the quality of picture in the output MPEG sequence. Since all packets are treated equally, this means that the packet loss rate would affect all the packets in the same manner. From Figure 53, it can be noticed that the NAK

requests would jump drastically when the 12.5%threshold is exceeded and they reach almost 40% when the packet loss rate is 15%. This huge rise would flood the network with never-ending retransmission requests. It is obvious that this method is not comparable to the layered approach, in fact, its performance is even worse than the pure ARQ systems.

3. At 15% packet loss rate, the retransmitted packet percentage is almost 11%, which is almost 27% of the requested packet by the receiver. Moreover, this percentage would not have been achieved without decreasing the transmission rate to 200 Kbps, which is very low.
4. The residual error and the throughput figures clearly show that this non-layered approach would divide the picture quality into two divisions: good or bad. Either the picture is perfectly reconstructed without any error, or the picture is purely bad and could not be comprehended. This is another disadvantage in comparison to the layered approach results.
5. Finally, it should be noted that this non-layered approach would introduce almost 16.8% overhead, which is considerably less than the 27% than that of the layered approach

In conclusion, the layered approach would behave more gradually and would greatly increase the level of picture quality, but on the cost of higher overhead and higher processing power needed in the receivers.

5.4 Further Enhancement to the Approach

One last enhancement on the proposed Layered Hybrid ARQ/FEC approach is to eliminate the retransmission requests for the B packets. That means that, even when a packet was not RS decoded properly, no NAKs are sent to the server. Instead, the client program would directly deliver the B packets *with* the errors to the playing buffer. It was noticed that the picture quality would not drop to a noticeable extent, since the residual error would not deviate much from the values got in Chapter 4 as it can be clearly seen from Figure 57. The main advantage of this enhancement is that the network would not endure the load of NAKs and retransmissions. This method would be very useful when the packet loss is bursty in nature, and it would be less effective when the packet loss is uniformly random. This is due to the fact, which was mentioned before, that concealing burst is easier than concealing random errors in MPEG players [22]. The screen shots below show a resulting Miss America sequence that endured 13 % random packet loss (at a transmission rate of 800 Kbps) and that no NAKs for the destroyed B packets were asked.

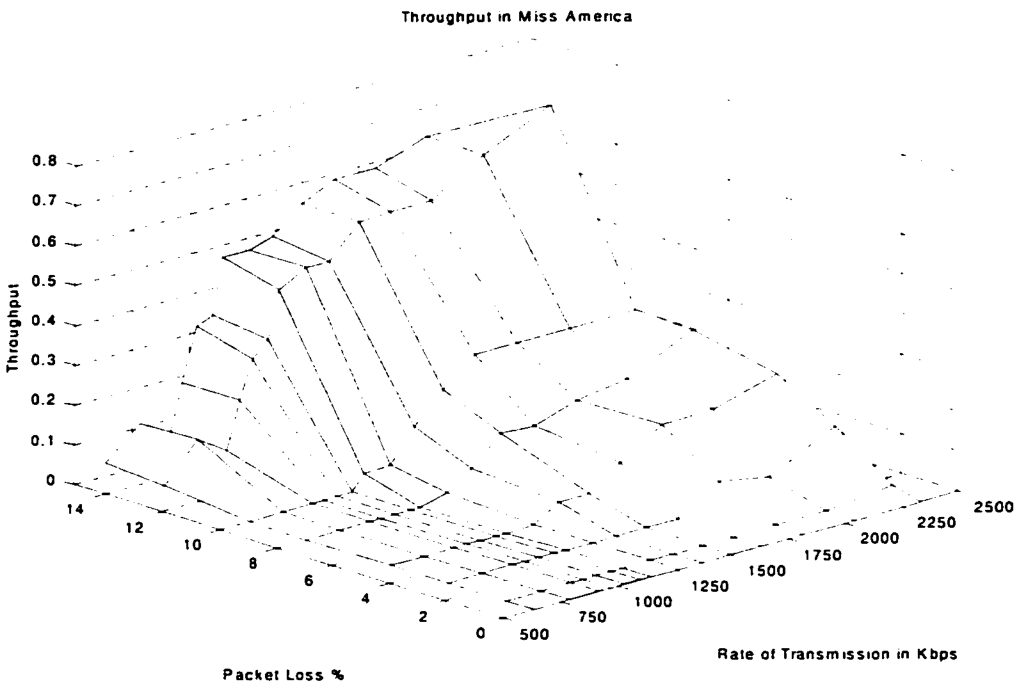


Figure 57: Throughput for Miss America without asking for B packets' retransmission, using layered approach

Frame 57

Frame 69

Figure 58: Some screen shots of Miss America, packet loss is 13%, without asking for B packets' retransmission

It is noticed that picture quality would be categorised as tolerable, just like when the same parameters are applied to a Layered Hybrid approach with B packets' retransmission. In some MPEG players (such as Xing and MTV), the difference would not be noticed between the two approaches, since one error concealment method - repeating the last good frame instead of the destroyed frame- would behave similarly in the two cases.

5.5 Comparing with Previous Similar Techniques

The aim of this section is to list, and compare when possible, some of the techniques that were studied and implemented before. These techniques have the same basic goal as the Layered approach; conveying the best QoS for MPEG streams by combining some techniques such as FEC, ARQ, information prioritisation and others. It should be mentioned that the Layered Hybrid ARQ/FEC, as it was described in this thesis, was not implemented before. Therefore, all the methods that are yet to be discussed will differ substantially from the Layered approach.

5.5.1 FEC-Based Video Streaming Using Pre-Interleaving

This method was introduced in [16], [17], and [18]. This techniques implements Reed -Solomon and interleaving as the basic coding algorithms in order to send/receive videos over lossy channels. The basic twist in this technique is that the interleaver block precedes the RS channel coding block, as it can be seen in Figure 59.

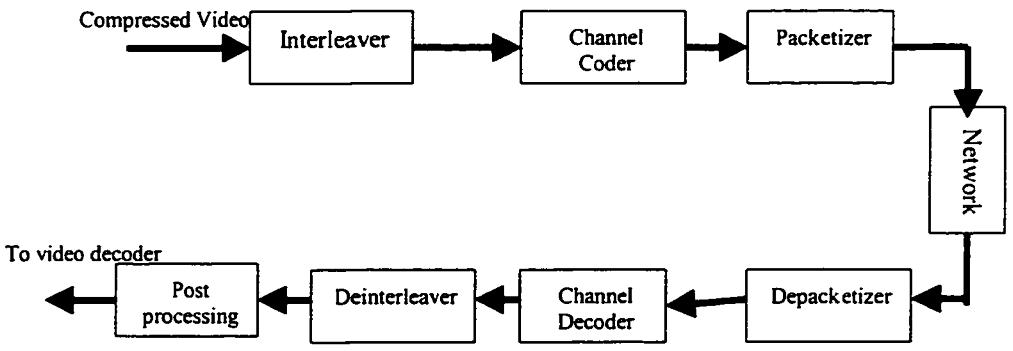


Figure 59: Block diagram of Pre-interleaving method [17]

The study justified this change as follows. As it is well known, MPEG players apply error-resilience and error concealment methods. It was shown in MPEG standard [17] that the decoders and players are more effective in handling bursty errors than random errors, for the same BER [17]. It is true that in conventional methods (where channel encoding precedes interleaving), the bursty error of the networks would be randomised over many packets which makes decoding easier. However, should any residual errors remain after RS processing, the quality of the picture would degrade significantly and the throughput would decrease. Using the pre-interleaving method, would re-convert the random errors into bursty errors which is desired pattern for the block-based video source decoding [16].

The experiments that were conducted during the study that was implemented in [17] showed that this approach would be superior to conventional approaches. The video sequence that was chosen is a 150-frames MPEG-4 "Foreman", with a frame rate of 15 frames/sec and a source bit rate of 100 Kbps, and there is 1 I Frame every 50 frames. The amount of redundancy chosen was 10 %, using a Reed Solomon with N=128. The packet loss chosen was bursty in nature (with burst length is 5 packets), and was ranged between 0 and 10%.

The PSNR (*Peak Signal to Noise Ratio*) of the Luminance (Y) factor, at packet loss of 10 % for the pre-interleaving method, had 5 dB gain over the conventional method [17], and as Figure 60 shows, the throughput had almost 30% improvement over conventional method. It can be seen that the throughput would fall to 90% when pre-interleaving is used at 10 % packet loss. Those results are comparable to the results of the Layered approach introduced in this thesis (Figures 47 and 48), though layered approach gives better performance. This approach, however, does not give any priority to frames.

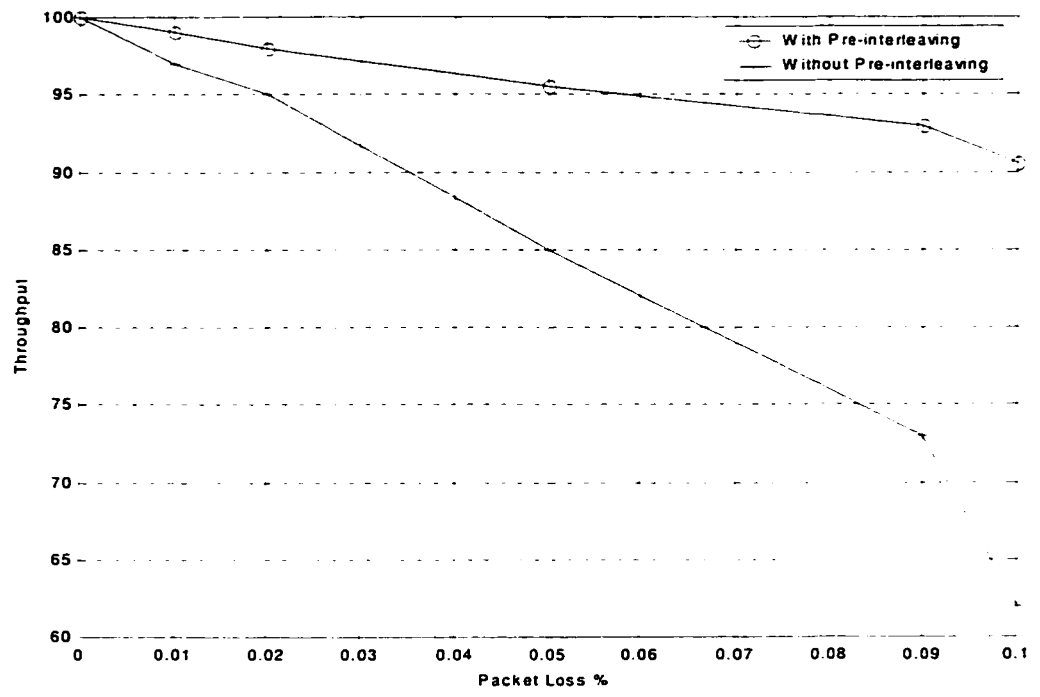


Figure 60: Throughput for pre-interleaving method [18]

5.5.2 Adaptive Hierarchical Coding Scheme

This method was introduced in [22]. The basics of this technique resemble to a large extent the layering principle upon which this thesis was based. The techniques can be reviewed in details in [22]; however, only its basics will be presented here.

In order to control the error propagation, the receiver cannot tolerate losing specific MPEG stream information, such as the DCT coefficients of blocks, motion vectors, relative addresses of MBs, ...etc. The proposed approach divides the MPEG bitstream into two partitions: *base layer*, which contains low-frequency coefficients, and *enhancement layer*, which contains high-frequency coefficients. If the parts of the enhancement layer are lost, they are simply replaced by zeros and the image is reconstructed using the base layer and the dummy enhancement layer, and the quality of image would be acceptable [22]. The base layer would contain all the headers, all the control information at the macroblock level, such as motion vectors, macroblock type, motion type, relative address of the macroblock in the slice, as well as the DCT coefficient of each block encoded as intra. It would also contain all the DCT coefficients different from zero up to the breaking point (which separates the two partitions). The remaining DCT coefficients different than zero, up to the end of the block (EOB), will make part of the enhancement layer. Other headers are also inserted in the enhancement layer, such as sequence, GOP, picture, slice and end-of-sequence headers [22]. This replication is the only overhead added to sequence.

This method introduced as much as 20% overhead to the total MPEG sequence Flower Garden. Changing the place of the breakpoint would change the overhead factor. The main aim of the study is to find the places of break points to compromise between the quality of image and the traffic generated.

The same source also introduced new algorithms for spatial and temporal error concealment for the MPEG player, in addition to building a packetization scheme for the two layers. This method was designed for VBR (Variable Bit Rate) -ATM, and the study would depend solidly on ATM standard. The new technique was compared with other methods and results were found. Since the experiments had criteria different of the criteria used in this thesis, the two results would not be compared. Neither FEC nor ARQ mechanisms were implemented in this method.

Other interesting methods to deliver MPEG video using combinations of methods (FEC, ARQ, Hybrid ARQ/FEC…) can be reviewed in details in [23], [24], and [25]. It should be noted that the specific combination of frame prioritisation with Hybrid ARQ/FEC was not implemented before.

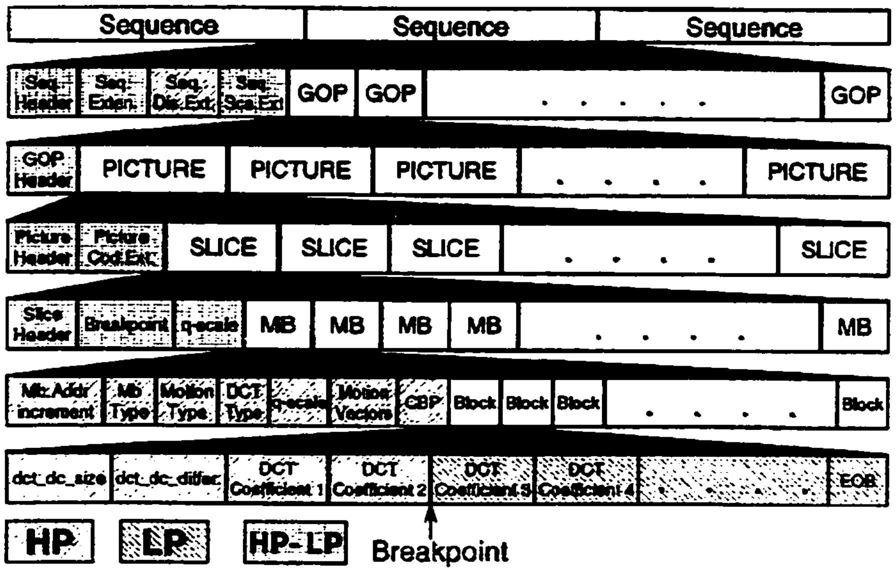


Figure 61: The partitioning scheme of Adaptive Hierarchical Coding Scheme [22]

CHAPTER 6

Conclusion, Limitations, and Future Work

6.1 Limitations of the Approach

Despite of all the advantages of using Layered Hybrid ARQ/FEC, there exist some limitations and drawback in the approach. The limitation factors in this approach are 1) the overhead and 2) the delay problems. Both problems are discussed below.

Overhead problem: The major problem in this approach, and in all approaches that applies FEC, is the *overhead* when neither errors nor packet losses occur in the channel during the communication sessions [7]. That means that more packets than needed are transmitted, more bandwidth is reserved, and more processing power is needed when no packet losses occur. In the actual experiments conducted, Miss America's MPEG sequence suffered 32% of overhead and Flower Garden had a 27% of overhead. In other words, in an error-free channel, 32 % of the packets and 27 % packets sent from the server to the client will no have any useful role. This problem can only become worse if the system was extended from unicast to multicast systems.

Another serious problem that is caused by overhead is the processing problem needed in the client. The processing power *available* at the client is directly related to the processing speed of its CPU. On the other hand, the processing power *required* in the client would be affected by the rate of transmission, BER and/or packet loss rate, and the rate of RS decoding. That means that if the RS decoder could not finish decoding the previous packet *before* the next window of packets arrives, the quality of the picture would degrade. The RS decoder that was used in the experiments can decode with a speed up to 1.5 Mbytes/sec when *k*=223, and up to 0.5 Mbytes/sec when *k*=191, when the number of erasures = *N* -*k* (on a Pentium II 700 MHz Machine) [26]. This RS decoder is not optimal and can be switched with other faster and more efficient decoder for better results (some decoders are optimised to achieve decoding speeds of 15-20 Mbytes/sec on Pentium I 133 machines). Processing problem will definitely limit the maximum transmission rate which is undesirable. Moreover, this problem would cause great deal of trouble in multicast sessions; every client has its own QoS requirements and processing power limitations and the server should take all these factors into account. Finally, RS decoding would be inefficient a undesirable in wireless systems since they consume much power, the thing that cannot be compromised in any design of wireless system

Another issue that is raised by the overhead problem is the meaningful number of bytes that are received for a specific transmission rate (*effective throughput rate*). As it was mentioned before, the layered approach would definitely introduce more overhead than the non-layered approach (27 % for layered while 16.8% for non layered Flower Garden). That means, if the transmission rate were 1 Mbps, the *meaningful* data bytes (packets excluding parity bytes) would be received at a rate of 730 Kbps for the layered approach, and at a rate of 840 Kbps for the non-layered approach. The difference is almost 11.5 % increment for the non-layered approach. In the Table 9, the maximum effective throughput rates (when the transmission rate is maximum) are shown for Flower Garden MPEG sequence, using both approaches.

Packet Loss %	Maximum Effective Throughput Rates for non-Layered Flower Garden (Mbps)	Maximum Effective Throughput Rates for Layered Flower Garden (Mbps)	Percentage of rate increment using non layered approach
1	2.352	1.33	0.76
3	2.1	1.3	0.61
5	1.974	1.095	0.8
8	1.512	0.92	0.64
10	1.26	0.708	0.78

Table 9: Comparison between maximum effective throughput rates for the Flower Garden sequence using layered and non-layered approaches

Needless to say that non-layered is superior in this case, as long as the packet loss rate is below the threshold of 12.5%. To be able to play real time (i.e. as soon as all processing of the first window at the receiver is finished), the effective throughput rate should be greater than the playing rate of the MPEG sequence. Since Miss America's playing rate is 500 Kbps (see Table 5), real time playing could be achieved in layered approach even when the packet loss rate is 10 % (since maximum effective throughput rate would be 800 Kbps). However, for the Flower Garden sequence, real time playing would not be efficient, since the playing rate of the sequence is 1.5 Mbps.

Overhead problem is not restricted to FEC, it can be also extended to ARQ also. In the proposed algorithm, if any packet could not be decoded, the client would immediately send the sequence number of the packet to the server asking for retransmission. This means that a whole TPDU, with all its UDP and IP headers [27], would be sent just to convey a 2-byte message to the server. At the same time, the server would send the 259-bytes packet as soon as it gets the NAK packet from the client, which also means that one TPDU is sent to convey 259-bytes message to the client. This problem is inevitable because speed of retransmission was the main factor that controlled the design of this approach. One last point should be mentioned is that the server in this algorithm should always keep two versions of the MPEG file, one RS encoded and one Interleaved, in order to process the NAK requests coming from the client. That means double hardware space should be allocated for every MPEG file to be sent. Again, this was a sacrifice that had to be done in order to deliver to the client a retransmitted packet that does need to be checked, the things that saves the client some processing.

Delay problem: Despite the great advantages of interleaving, which were mentioned in section 3.4, its implementation would introduce a specific delay. This delay comes from the fact that at least *kmax* packets should be received by the client in order to have meaningful data. RS decoding and interleaving would introduce a substantial delay effect to the process. RS process would delay the deliverance of a specific packet till it finishes decoding, and interleaving would delay the start of decoding a window till all packets (or at least k_{max}) of one window are received. This delay would be added to the frames' reordering delay [2] of the MPEG player and may degrade the efficiency. At the maximum transmission rate, the de-interleaving delay would become very close to the RS decoding time of one window, since the time of sending one window is just enough to decode the previous one. Table 10 shows the delays that would be introduced due to interleaving at maximum transmission rates.

Packet Loss %	Delays due to interleaving in non-Layered Flower Garden, at maximum transmission rate (milliseconds)	Delays due to interleaving in Layered Flower Garden , at maximum transmission rate (milliseconds)	Delays due to interleaving in Layered Miss America, at maximum transmission rate (milliseconds)
1	189	291	300
3	212	296	353
5	225.7	353	378
8	294.6	420	408
10	353	546	442

Table 10: Comparison between delays due to interleaving and decoding for the Flower Garden using layered and non-layered approaches and Miss America using layered approach

The values in Table 10 would indicate the minimal value of delay that must be present at startup before the playing of the MPEG video begins. If the first window delivered to the client contains enough frames for the reordering process, then the startup delay would be:

Startup delay = Deinterleaving delay + Decoding delay + Reordering delay

When the transmission delay is maximal, Decoding delay can be approximated to the Deinterleaving delay. So :

Startup delay = 2xDeinterleaving delay + Reordering delay

When the MPEG sequence has a GOP of 2 consecutive B frames, the reordering delay can be approximated by 7 frames. This means that the reordering time would be 7/30(frames per second)= 233 ms. Interactive audio-visual applications, such as video conferencing, cannot tolerate delays more than few hundred milliseconds [28]. On the other hand, non-interactive applications, such as video on demand, can tolerate delays up to 1 second or more [29]. Typical values of delay requirements for audio/visual applications are [29]:

- Tight (up to 200 ms)
- Medium (up to 500 ms)
- Loose (more than 500 ms)

Setting these values as constraints and comparing with the values of Table 10, it can be noticed that the layered approach implemented, as it was implemented during the experiment, would not fit to interactive applications (tight constraints). However, it would fit for other applications that do not impose tighter delay constraints.

Jitter, on the other hand, would not impose a great problem, like the delay, in the proposed approach. That is because of the fact that the client program would be working on one window while receiving the packets of the next window. Any out of order packets can be rearranged in the client according to their 2 bytes sequence number. Moreover, if some packets suffered from substantial delays on their routing path to the client, they would be considered missing and decoding might be needed to recover them.

6.2 Future Work

Although that the previous section drew a dark image about the deficiencies of the proposed approach, almost all the problems mentioned could be solved with ease, and the great advantages of this approach would appear easily on the surface.

The first enhancement for the approach should be adding flow control mechanisms. Flow control algorithms would limit the residual error, since it decreases the number of packets sent per second when the receiver is not able to process the packets as fast as they are sent. Moreover, flow control would give more precise results, since the program would crash several times when the transmission rate is above maximum transmission rate *and* packet loss rate is higher than 12.5 %.

Many of the various problems mentioned above could be solved when a faster and more powerful RS decoder is used. As it was stated before, some software decoders are optimised to achieve decoding speeds of 15-20 Mbytes/sec on Pentium I 133 machines [26]. That would increase enormously the maximum transmission rate that can be achieved for any packet loss error, which would increase the effective throughput rate, which would allow MPEG real time playing.

Another important factor that would affect to a noticeable extent is the parameters of the RS decoder. Changing the values of k_I, k_P, and k_B would reduce the overhead substantially. In the experiments conducted in this thesis, the values of k_I, k_P, and k_B were not chosen to give optimal performance. Moreover, it can be noticed, that 191 and 207 for k_I and k_P are not realistic in networks (e.g. internet) since 25% and 18.75% packet losses do not usually occur in networks, unless there is something unusual. In [14] the overhead was computed optimally for each frame type. However, these values were only theoretical and it was stated that this optimality could be hard to achieve, since the packet loss distribution should be determined for each video stream and the current traffic load. This might be unfeasible especially in real-time applications [14]. One way to avoid dynamic packet loss in channels is to use dynamic FEC for all the three frames; when the receiver senses that many packets are lost, it could initiate a warning to the server so it can raise the level of priority to the important frames at the time of encoding. This should be investigated thoroughly, taking into consideration multicasting issues.

Another way to reduce the overhead in the network and the delay is to try to switch from Hybrid Type I to Hybrid Type II [12]. That way the sender would send only the data part to the receiver and if some packets are missing, the server would send the parity part. This would take some further manipulation in the shape of RS decoding so the data part is encoded in separate packets from the parity ones.

To reduce the delay more, the interleaving depth could be decreased in the approach used. Decreasing the interleaving depth has two advantages.

1. It would reduce the waiting time for delivering data to the playing buffer. It should be noted that down sizing the interleaving depth would affect on delay only when the parameters of the RS encoder/decoder are changed. That is because as N reduces, the decoding time reduces for the packet and it can be delivered

faster to the playing buffer. However, if the parameters of the RS encoder/ decoder are kept the same as before (N=255 and k_I, k_P, and k_B), this downsize would not affect very much on the delay. That is because the reordering process needs a specific number of frames to be already delivered so it can start. With smaller interleaving depth, fewer frames would be delivered to the playing buffer and the MPEG player should wait for another window before starting to reorder the frames.

2. Reducing interleaving depth would definitely reduce the number of packets that might need to be decoded at the receiver. As it was stated in Chapter 4, if the lower part of the sequence number of at least one lost interleaved packet is smaller than k_I, *all* the packets in a window must be decoded. If the depth were reduced, one interleaved packet would not contain bytes from all encoded packets. Hence, one or more packets could be lost without affecting all packets in a window, and this reduces decoding power needed in the receiver.

To reduce delay more and be able to play interactive video (e.g. video conferencing applications), another point should be taken into consideration; reordering time. Using a GOP that does not contain any B frames can eliminate this reordering time. In this case, the frame playing order would be the same of frame encoding order and that would eliminate the 233 ms of delay.

UDP is the transport protocol that is used in these experiments. UDP header has its own checksum [27], and if any error is detected at the receiving end, the packet would be discarded [30]. That makes UDP not suitable for wireless applications, since the BER in wireless channels is high, and so many UDP packets would be discarded [30]. If the Layered Hybrid ARQ/FEC approach is to be applied in an all-IP wireless system, this factor should be taken into consideration. In [30], a new modified UDP protocol was proposed. This new protocol, which was called Complete UDP (CUDP), would be able to identify the damaged packets in the TPDU based on lower layer information and, instead of discarding the whole TPDU, would pass the damaged packets to FEC decoder. In the simulations, this CUDP was proven to be superior to UDP when combined with interleaving and FEC even when the packet loss is in the order of 10% [30].

Another point to consider is the subjective methods that can be used for subjective evaluations of the quality of images got at the receiver. As mentioned in section 5.1, PSNR and MLD are different methods, other than residual error, that give more precise idea about the quality of received MPEG video. Subjective evaluation manners are being investigated and PSNR was found to be an excellent subjective evaluation. PSNR role should be investigated thoroughly in the future.

Finally, shifting to hardware, instead of software, should be considered. That shift would boost the performance of the system, since hardware VLSI calculations would take much less time than application-level software. Using hardware, all operations would finish faster (encoding/decoding, interleaving/deinterleaving, ...etc), and hence interactive real time multimedia applications might be feasible to implement. The feasibility of this shift should be investigated in the future.

6.3 Conclusion

To wrap up this thesis, some concluding remarks will be stated. This thesis presented a genuine implementation of a Layered Hybrid ARQ/FEC client/server system using C language. The main aim of this idea was to enhance the QoS for

MPEG video streams' transmission. The approach made use of the different types of frames in the MPEG sequence and the importance of each for the quality of picture. I Frames are the reference frames and are the most important ones, since any error affecting them could affect another frames. P Frames are less important than I Frames and B frames are the least important. This property suggests that to achieve best QoS while transmitting via a lossy channel, each type of these frames could be given different priority according to its importance. The priority that was implemented is giving each type of frame different level of Reed Solomon encoding. That means $k_B > k_P > k_I$, for the same N. That is the basic idea of layered FEC. In the experiments conducted, the values were $k_B = 223$, $k_P = 207$, $k_I = 191$, and N=255 where the symbol is one byte.

This Layered FEC was integrated also with Selective Repeat ARQ method in order to create the Layered Hybrid ARQ/FEC Type I Approach. Any packet that could not be decoded correctly due to extensive packet loss will be retransmitted to the receiver. Finally, interleaving was implemented as the last stage of channel encoding.

Real tests were conducted on this Layered Approach and satisfactory results were obtained. These results clearly show that using layered ARQ/FEC approach would be superior to non-layered approach. The non-layered ARQ/FEC approach would segment the quality of picture into two regions: good or bad. That means that either the picture would be perfect and error-free, as long as the number packets lost in an interleaved window are below N - k_B, or the picture is bad, as long as the number packets lost in an interleaved window are greater than N - k_B. On the other hand, when the layered approach is implemented, the quality of picture degrades gradually from perfect, then acceptable, then tolerable and finally bad, even when the number packets lost in an interleaved window are greater than N - k_B.

It was noticed during the experiments that this approach is also superior to pure ARQ systems. In the case of pure ARQ system, the amount of retransmission requested would directly be related to the amount of packet loss. On the other hand, when the new approach is applied, the amount of retransmissions requested would be noticeably less due to the FEC decoding at the receiver. Hence, the amount of network loading would be significantly less.

The results of Layered Hybrid ARQ/FEC approach were also compared with other methods and techniques that were applied in previous experiments. Although that Layered Hybrid ARQ/FEC algorithm was not implemented practically before, some techniques discussed behave similarly to this approach and they have the same goal; trying to achieve the best QoS for MPEG files without overloading the network with retransmissions.

The disadvantages and limitations of this approach were also taken into consideration. Overhead is the major disadvantage of this approach, because when the channel is error-free, the RS parity bytes would be considered as a burden on the network and would not have any use. Also, start up delay due to interleaving and RS decoding would also be significant. These disadvantages, however, could be eliminated eventually in the future. Using faster RS encoders/decoders, optimising the values of k_B , k_P , and k_I , changing interleaving depth, using flow control methods, and dynamic layered encoding could reduce the overhead and delay significantly.

The Layered Hybrid ARQ/FEC approach for MPEG communication, or even for other applications, should be very promising for future research. Any applications, especially interactive multimedia applications, which contain different information that differ in importance would easily make use of this approach. This makes Layered Hybrid ARQ/FEC approach feasible to implement in video conferencing, video and demand, satellite communications, cable video, and many other entertainment applications.

References

[1] P.N. Tudor, " Tutorial, MPEG-2 video compression", IEEE Electronics & Communication Engineering Journal, December 1995

[2] Barry G. Haskell, Atul Puri and Arun N. Netravali, "Digital Video: An Introduction to MPEG-2", Chapman & Hall, November 1996

[3] Thomas Sikora, " MPEG Digital Video-Coding Standards", IEEE Signal Processing Magazine, September 1997

[4] Stephen Gringeri, Bhumip Khasnabish, Arianne Lewis, Khaled Shuaib, Roman Egorov, and Bert Basch, " Transmission of MPEG-2 Video Streams over ATM", IEEE Multimedia, January-March 1998

[5] Olivier Verscheure and Xavier Garcia, " User-Oriented QoS in Packet Video Delivery", IEEE Network, November/December 1998

[6] Joan L. Mitchell, " MPEG Video Compression Standard", Chapman & Hall International Thomson Publishing, 1996

[7] Hang Liu, Hairuo Ma, Magda El Zakri, and Sanjay Gupta, "Error control schemes for networks : An overview", Mobile Networks and Applications 2, 1997

[8] Elwyn R. Berlekamp, Robert E. Peile, and Stephen P. Pope, " The Application of Error Control to Communications ", IEEE Communications Magazine, April 1987

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

[10] Shu Lin, Daniel J. Costello. Jr., and Micheal J. Miller, " Automatic-Repeat-Request Error Control Schemes", IEEE Communications Magazine December 1984

[11] John B. Anderson and Seshadri Mohan," Source and Channel Coding: An Algorithmic Approach", Kluwer Academic Publishers, 1991

[12] Stephen B. Wicker and Vijay K. Bhargava (editors), " Reed Solomon Codes and their Applications ", IEEE Press, 1994

[13] I. E. G. Richardson and M. J. Riley, " MPEG Coding for Error-Resilient Transmission ", Image Processing and its Applications, 4-6 July 1995, Conference Publication No. 410, IEE 1995

[14] Rainer Storn, " Modelling and Optimisation of PET-Redundancy Assignment for MPEG Sequences ", ICSI Publication, May 1995

[15] Jian Zhang, Micheal R. Frater, John F. Arnold, and Terence M. Percival, " MPEG-2 Video Services for Wireless ATM Networks", IEEE Journal of Selected Areas in Communications, Vol. 15, NO.1 January 1997

[16] Jianfei Cai and Chang Wen Chen, " Use of Pre-interleaving for Video Streaming Over Wireless Access Networks", Proceedings 2001, International Conference on Image Processing. Vol. 1, 2001

[17] Jianfei Cai and Chang Wen Chen, "FEC-Based Video Streaming over Packet Loss Networks with Pre-Interleaving", Proceedings 2001, International Conference on Information Technology, Coding and Computing

[18] Jianfei Cai and Chang Wen Chen, "Video Streaming: An FEC-Based Novel Approach", Canadian Conference on Electrical and Computer Engineering. Vol.1 2001, Pages: 613-618

[19] Peng Ge and Philip K. McKinely," Comparisons of Error Control Techniques for Wireless Video Multicasting", 21st IEEE International Performance, Computing, and Communications Conference, 2002. Pages 93-102

[20] H. Linder, I. Miloucheva, and H. D. Clausen, "A Forward Error Correction Based Multicast Transport Protocol for Multimedia Applications in Satellite Environments", IEEE International Performance, Computing, and Communications Conference, 1997. Pages 419-425

[21] Dong H. Chen, "Adaptive FEC/ARQ Schemes for Stream Media Multicast over the Internet", Thesis in the department of Electrical and Computer Engineering, May 2002

[22] Pedro Cuenca, Luis O. Barbosa, Francisco J. Quiles, and Antonio Garrido," Loss-Resilient ATM Protocol Architecture for MPEG-2 Video Communications", IEEE Journal on Selected Areas in Communications, Vol. 18, NO.6, June 2000

[23] Pascal Frossard and Olivier Verscheure, "AMISP: A Complete Content-Based MPEG-2 Error-Resilient Scheme", IEEE Transaction on Circuits and Systems for Video Technology, Vol. 11, NO.9, September 2001

[24] Injong Rhee and Srinath R. Joshi, "Error Recovery for Interactive Video Transmission over the Internet", IEEE Journal on Selected Areas in Communications, Vol.18, NO.6, June 2000

[25] Daniel Grobe Sachs, Igor Kozintsev, and Minerva Yeung, "Hybrid ARQ for Robust Video Streaming over Wireless LANs", Proceedings. International Conference on Information Technology: Coding and Computing, 2001. Pages:317-321

[26] http://www.ka9q.net/, Phil Karen, Reed Solomon programs in C/Assembly, July 1997

[27] William A. Shay, "Understanding Data Communications & Networks", International Thomson Publishing Company, Second Edition 1999

[28] E. Kelmmer," Subjective Evaluation of Transmission Delay in Telephone Conversations", Bell Sys. Tech. J., Vol.46, July 1967, Pages: 1141-1147

[29] Georg Carle and Ernst W. Biersack, "Survey of Error Recovery Techniques for IP-Based Audio- Visual Multicast Applications", IEEE Network, November/December 1997

[30] H. Zheng and J. Boyce," Packet Coding Schemes for MPEG Video over Internet and Wireless Networks", IEEE Wireless Communications and Networking Conference September 2000, NO. 1, Pages 191-195

[31] Raj Talluri, " Error-Resilient Video Coding in ISO MPEG 4 Standard ", IEEE Communication Magazine, June 1998, Pages 112-119

Appendix

Pseudo Code for the Server and Client Programs for Layered Hybrid ARQ/FEC

A.1 Server Side

A.1.1 Offline Encoding

/*Read the MPEG sequence byte by byte to discover the prefix 0x0001 and the start codes for different frames */

```
read_next_byte();                        /* read first byte from file */
Type_found =test_header();               /* test if prefix found*/
while (! EOF){                           /* While Not End of File */
switch (Type_found){
        case B_Type:
        K = KKB;                         /* KKB= 223 */
        break;

        case P_Type:
        K = KKP;                         /* KKP= 207 */
        break;

        case I_Type:
         K = KKI;                        /* KKI= 191 */
         break;

        case H_Type:                     /* Sequence header */
        K = KKI;
        break;
} /* close switch */
```

```
/* time to fill the data part of the packet according to K */



while(!Type_found=test_header && ! EOF){        /* as long as next frame is not
                                                detected   */
  if(Data_counter <K-6)                         /*K-6 to make include 4 header
                                           bytes and 2 CRC bytes */
      {
           Data_part[ Data_counter]=read_next_byte();  /* fill the data part of the
                                                            packet */
           Data_counter++;
      }                                         /* close if */
  else                                          /* the data part completely filled
                                                */
       {

           Data[ ]=Add_headers(Data_part[ ]);  /* Add 1 byte Type field, 2 bytes
                                                Sequence number, and 1 byte
                                                Data length */
           Data_with_CRC [ ]= CRC16(Data,K);    /* Add 16 bits CRC at the end
                                                    */
           Packet[ ]=RS_Encode(Data_with_CRC[ ], K);
           Packet [ ]=Add_dummy(Packet);
           encoded_file.write(Packet, 256);      /* write the resulting 256 byte in
                                                the encoded_file */
         }                                      /* close else */
}                                               /*close while next frame not
                                                detected */
if ( Data_counter <K-6 && ! EOF)                /* next frame detected*/
  {
           Data=Add_padding(Data);
```

```
            Data[ ]=Add_headers(Data_part[ ]); /* Add 1 byte Type field, 2 bytes
                                                  Sequence number, and 1 byte
                                                  Data length */
            Data_with_CRC [ ]= CRC16(Data);   /* Add 16 bits CRC at the end */
            Packet[ ]=RS_Encode(Data_with_CRC[ ], K);
            Packet [ ]=Add_dummy(Packet);
            encoded_file.write(Packet, 256);   /* write the resulting 256 byte in
                                                  the encoded_file */
        }                                      /* close if next frame detected */
}                                              /* close while(!EOF) */
interleaved_file = interleave(encoded_file,interleaving_depth);
                                               /* interleaving_depth =256*/
```

A.1.2 Sending the Packets

```
/* Main Transmission Process */
interleaved_file.open();
while(!EOF)                                    /* while not end of file */
{
      fill_window_tosend();                    /*load 256 interleaved words into
                                                 window to be sent */
      save_last_interleaved_window;            /*in case retransmission of
                                                 interleaved packets is requested*/
      save_last_fresh-window;                  /*in case retransmission of fresh
                                                 (only RS encoded) packets is
                                                 requested*/
      while(counter1 < WINDOWSIZE)             /* while the 256 are not sent
                                                 completely*/
      {
         while (counter2 < TPDULENGTH)         /* attach some packets in one
                                                 TPDU to be sent to UDP */
```

```
  {
     memcopy(TPDU_tobe_send, Packet[counter1])
     counter2++;
     counter1++;
  }
delay_in_micro_seconds(RATE);          /* wait for RATE in
                                       microseconds/*
sendto(socket,TPDU_tobe_send,TPDULENGTH*WORDLENGTH)
                                       /* send to UDP socket which
                                       sends the TPDU to the receiver */
 counter2=0;
 if (NAK_Fresh_Flag!=0)                /* NAK Fresh detected */
   {
   while(number_of_NAK_requested !=0)
    {
       if(Seq_number_still_in_window)
          {
           memcpy(Retransmission_TPDU,packet_requested)
           number_of_retransmissions++;
          }
        else discard_for_timeout();
        number_of_NAK_requested--;
      }/* close while: number_of_NAK_requested =0*/

     sendto(socket,Retransmission,
                 number_of_retransmissions*WORDLENGTH)
                                      /* send Retransmission the socket
                                      which sends the TPDU to the
                                      receiver */
         NAK_Fresh_Flag=0;
      }                               /* close if NAK_Fresh_Flag */
```

```
            if (NAK_Interleaved_Flag!=0)        /* NAK Interleaved detected */
             {
              while(number_of_NAK_requested !=0)
               {
                 if(Seq_number_still_in_window)
                    {
                     memcpy(Interleaved_Retransmission_TPDU,packet_requested)
                     number_of_retransmissions++;
                    }
                  else discard_for_timeout();
                  number_of_NAK_requested--;
                }/* close while: number_of_NAK_requested =0*/

               sendto(socket,Interleaved_Retransmission,
                          number_of_retransmissions*WORDLENGTH)
                                                /* send Interleaved
                                                Retransmission the socket which
                                                sends the TPDU to the receiver */
                  NAK_Interleaved_Flag=0;
                }                               /* close if NAK_Interlaved_Flag
*/
            }                                     /* window is sent completely :
                                                close while(counter1 <
                                                WINDOWSIZE) */
}                                               /*close while (!EOF)
```

A.2 Client Side

A.2.1 Main Processing Loop

/* Process the packets in previous window while the current window is being filled */

```
if(number_of_packet_received[0]==0)            /* if the interleaved packet
                                               number 0,which contains the
                                               Type field values of all the
                                               packets in the processed window,
                                               without it no packet can be
                                               decoded correctly */
{
   send_NAK_intereleaved();                    /*ask for retransmission of
                                               interleaved packet number 0*/
    wait(T);                                   /*wait some time for
                                               retransmission */
     if(number_of_packet_received[0]==1)       /* packet retransmitted and
                                               received */
     {
       deinterleave();                         /* deinterleave this packet */
     }
else                                           /* packet was not received */
      {
        deliver_all_packets();                 /* deliver all packets with
                                               possibility of errors*/
        All_Flag=1;                            /* flag to indicated that all
                                               packets were delivered ,default
                                               value = 0*/

      }
```

```
}                                              /*close if
                                               (number_of_packet_received[0]=
                                               =0)*/
if(All_Flag=0)                                 /*start processing normally */
   {
     fill_erasure(number_of_packet_received[ ]); /*get the position of erasures
                                               according to Seq. numbers
                                               received   */
    while(packets_delivered<WINDOWSIZE)
     {
       K=test_header(word_tobe_decoded)        /* Read the Type field of the
                                               encoded word (packet) */
       CRC16(word_tobe_decoded,K)              /*test CRC for packet */
        if(correct_CRC)
          {
           deliver_packet();                   /* no error detected, deliver */
          }
       else                                    /*CRC not correct */
          {
          RS_Decode(word_tobe_decoded,K);
          if(correct_decoding)  deliver_packet();/*correctly RS decoded*/
          else                                 /*fail to decode*/
              {
             send_NAK_Fresh();                 /*ask for retransmission of this
                                               encoded packet*/
              wait(T);                         /*wait some time for
                                              retransmission*/
             if(retransmission_fresh_flag==1)/* retransmission flag*/
              { if(packet_was_requested)
                     deliver();               /*deliver unchecked */
               else {                         /*packet was not requested*/
```

```
                    dropped_counter++;          /* for calculating
                                                dropped/delivered with errors
                                                packet*/
                        }
                    }
                  else{                         /* retransmission not received */
                     deliver();                 /*delivered with possible errors*/
                     dropped_counter++;}
                    }                           /* close fail to decode*/

            }                                   /*CRC not correct*/
             packets_delivered++;
          if(need_to_flush_flag=1){             /* next window detected*/
          deliver_rest_of_the_window();         /*with possible errors*/
          All_Flag=1;
          update_windows_parameters();
          empty_temporary_buffer();
          break;}

          }                                     /*close while*/
          All_Flag=1;

     }                                          /*this window is finished and
                                                waiting for next window to
                                                process*/
```

A.2.2 Filling Current Window Process

```
/*This process interrupts the previous one, whenever a TPDU is detected in
 the UDP socket*/
recvfrom(socket,temporary_buffer,...)           /*TPDU detected in socket*/
test_header_of_interleaved_packet();
```

```
if(current_window)                                  /*packet belong to current
                                                    window*/
{
        deinterleave();
}

if(previous_window)
{
        deinterleave_in_previous_window();
}

if(next_window)
{
   if(All_Flag==1);                                 /*next window detected and all
                                                    packets in previous window
                                                    delivered*/
      {
         update_windows_parameters();
         deinterleave();

      }
   else {
       need_to_flush_flag=1;                        /* set this flag if some packets in
                                                    the previous window were not
                                                    delivere*/
       buffer_packet_temporarly();}                 /*put this packet in temporary
                                                    buffer*/
}

if(retransmitted_fresh)
{
  retransmission_fresh_flag==1;
```

```
   retransmission_counter++;
}

if(retransmitted_interleaved & previous_window)
{
   deinterleave();
    retransmission_counter++;
}

else

dropped_counter++;
```