The Appropriate FEC Parity Amount to Guarantee High Quality for Video Streaming

Mahmoud Ali Al-Shugran

UNIVERSITI UTARA MALAYSIA
2009

The Appropriate FEC Parity Amount To Guarantee High Quality for Video Streaming

This thesis is presented to the Graduate School
In fulfillment of the requirements for
Master of Science (Information and Communication Technology)
Universiti Utara Malaysia

By

Mahmoud Ali Al-Shugran
(801359)

KOLEJ SASTERA DAN SAINS
(College of Arts and Sciences)
Universiti Utara Malaysia

PERAKUAN KERJA KERTAS PROJEK
(Certificate of Project Paper)

Saya, yang bertandatangan, memperakukan bahawa
(I, the undersigned, certify that)

MAHMOUD ALI IBRAHIM AL-SHUGRAN
(801359)

calon untuk Ijazah
(candidate for the degree of) **MSc. (Information Communication Technology)**

telah mengemukakan kertas projek yang bertajuk
(has presented his/ her project paper of the following title)

THE APPRORIATE FEC PARITY AMOUNT TO GUARANTEE HIGH QUALITY FOR VIDEO STREAMING

seperti yang tercatat di muka surat tajuk dan kulit kertas projek
(as it appears on the title page and front cover of project paper)

bahawa kertas projek tersebut boleh diterima dari segi bentuk serta kandungan dan meliputi bidang ilmu dengan memuaskan.
(that the project paper acceptable in form and content, and that a satisfactory knowledge of the field is covered by the project paper).

Nama Penyelia Utama
(Name of Main Supervisor): **ASSOC. PROF. DR. SUHAIDI HASSAN**

Tandatangan
(Signature) : [illegible]

Tarikh
(Date) : 10/5/2009

PERMISSION TO USE

ABSTRACT

Efficiency and capability of forward error correction (FEC) mechanism to recover from losses, is the main idea behind its utilization in Real-time applications like video streaming over Internet. The number of Internet users continue to increase every day, thus network traffic will increase. Such increasing in network traffic will cause network congestion to increase, increasing in network congestion make data loss obvious problem that affect streaming video presentation quality.

The efficiency of FEC mechanism come from the reduction of the time needed to recover lost packets, and there is no need for an additional channel for retransmission. The goal of FEC is to add redundancy that can be used to recover from packet loss in multimedia application, this big task for FEC comes with main condition; to achieve acceptable and the desired quality of presentation, FEC has to be introduced with minimum overhead for real-time application.

However, depending on the fact that packet loss is often unknown and time varying. So, the amount of extra data that will be sent with the original data in the FEC block is predetermined comparing to the loss, hence loss recovery rate depends on the amount of redundancy data. For this reason FEC may take three characteristics depending on the amount of redundant data, *effective* (if the redundant is sufficient to the lost data), *ineffective* (if the redundant is too little to the lost data), or *inefficient* (if the redundant is too much to the lost data).

Our research will concentrate on this problem to propose a suitable solution, by matching the appropriate redundancy that can achieve effectiveness, through sending the proposed FEC parity amount aside with the original data of video application. Proposed results will guarantee minimum packet loss and packet loss ratio in streaming application for video.

Furthermore, through this study we will verify the affect of variance amount of redundancy data on the loss recovery rate, to find the appropriate parity amount, which can be effective in the presence of data loss. During that, Simulation results will demonstrate our progress for this project step by step to insure the works. Through extensive simulation, we will evaluate the scalability of the chosen redundancy data amount to improve aggregate FEC performance through video transmission.

ACKNOWLEDGMENTS

I would like to state my sincere appreciation to the following people, who have given me support and encouragement during this research:
My beloved father and beloved mother for all the guidance and support from them.
Furthermore, my special thanks to my supervisor Assoc. Prof. Dr. Suhaidi Hassan for all the advice, encouragement and supervision.
My appreciation to all the ns-2 users who have shared all the knowledge and opinions, last but not the least, to all my friends who supported me.

CONTENTS

Permission to Use i

Abstract ii

Aknowledgmentes iii

Contents iv

List of Figures viii

List of Tables ix

List of Abbreviation x

CHAPTER 1

INTRODUCTION

1.1 Introduction 1

1.2 Background 1

1.3 Problem Statement 5

1.4 Goals of the Study 6

1.5 Objectives of the Study 7

1.6 Research Questions 7

1.7 Significance of the Study 7

1.8 Scope of the Study 8

1.9 Report Structure 8

1.10 Summary 10

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction 11

2.2 Video Streaming Delivery Mechanism 12

2.3 Major Problem in Video Streaming 13

2.4 Bandwidth Reservation 13

2.5 End-to-end delay 14

2.6 Packet loss 14

2.7 Video Streaming and Packet Loss 15

2.8 Reason behind packet loss 16

2.9 How video streaming is affected by packet loss 16

2.10 How to recover lost packets 17

2.11 Forward Error Correction (FEC) 17

2.12 Advantages and Disadvantages of FEC 18

2.13 Major Problems with FEC and FEC Parity Amount 19

2.14 FEC Mechanism 20

2.15 Error-correcting code (ECC) 21

2.16 Proposed solution 24

2.17 Summary 25

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction 26

3.2 Network simulator 28

3.3 Experimental Objectives 29

3.4 Simulation Models and Environment 30
- 3.4.1 Simulation topology 30
- 3.4.2 Traffic characteristics 31
- 3.4.3 Simulation Parameters 32
- 3.4.3.1 Queuing Management Policy 32
- 3.4.3.2 FEC Setting 32
- 3.4.3.3 Bottleneck Bandwidth 33
- 3.4.3.4 Bandwidth 33
- 3.4.3.5 Delay 33
- 3.4.3.6 Simulation Data 34

3.5 Performance Metrics 34
- 3.5.1 Packet loss 34

3.5.2 Packet loss ratio (PLR) 35

3.6 Simulation Phase 35

3.7 Simulation Process 35

3.8 Results Analysis and Data Interpretation 36

3.9 Results Presentation 36

3.10 Summary 37

CHAPTER 4

SIMULATION PHASE AND PROCESS

4.1 Introduction 38

4.2 FEC Performance 39
4.2.1 FEC Performance Metrics 39
4.2.2 FEC Parity Amount 40

4.3 Experiment 42
4.3.1 Experimental Environment 42
4.3.2 Experiment Setup 42
4.3.3 Experiment Parameters 45
4.3.4 Experiment Execution 45

4.4 Simulation Results 46

4.5 Conclusion 56

CHAPTER 5

RESULT ANALYSIS AND PRESENTATION

5.1 Introduction 57

5.2 Results Analysis and Data Interpretation 57
5.2.1 Packet Loss 57
5.2.2 Results Validity 60
5.2.3 Packet Loss Ratio 64

5.3 Presentation of results 67

5.4 Experiment Conclusion 71

5.5 Summary 72

CHAPTER 6

CONCLUSIONS AND RECOMMENDED FURTHER STUDY

6.1 Introduction 73

6.2 Findings 73

6.3 Problems and Limitations 74
    6.3.1 Problems 74
    6.3.2 Limitations 74

6.4 Contributions 75

6.5 Future Works 75

References 76

Appendices 81

List of Figures

FIGURE 1.1 Video Streaming Over Internet 3

FIGURE 2.1: Block Encoding For Error Correction 22

FIGURE 2.2 Digital Communications System 22

FIGURE 3.1: Simulation Phases 27

FIGURE 3.2: Simulation Topology 31

FIGURE 4.1: Lost Packets At The First Router 40

FIGURE 4.2: Sent And Received Packets 41

FIGURE 4.3: Simulation Topology 43

FIGURE 4.4: Network Topology 44

FIGURE 5.1: Redundant Packet Vs. Packet Loss 69

FIGURE 5.2: Redundant Packet Vs. Packet Loss Ratio 71

List of Tables

TABLE 4.1: Simulation Parameters 45

TABLE 4.2: Fields In The Raw Trace-File 47

TABLE 4.3: Symbols In The Raw Trace File 48

TABLE 4.4: Results Of Simulation Using 5 Packets As Redundancy 49

TABLE 4.5: Results Of Simulation Using 10 Packets As Redundancy 50

TABLE 4.6: Results Of Simulation Using 15 Packets As Redundancy 51

TABLE 4.7: Results Of Simulation Using 20 Packets As Redundancy 52

TABLE 4.8: Results Of Simulation Using 25 Packets As Redundancy 53

TABLE 4.9: Results Of Simulation Using 30 Packets As Redundancy 54

TABLE 4.10: Results Of Simulation Using 35 Packets As Redundancy 55

TABLE 5.1: Lost Packets And Received Packets 58

TABLE 5.2: The Average Of Lost Packet And Received Packet 59

TABLE 5.3: The Confidence Interval Of Lost Packets 62

TABLE 5.4: The Average Of Lost Packets And Margin Of Error 62

TABLE 5.5: The Confidence Interval Of Received Packet 63

TABLE 5.6: The Average Of Received Packets And Margin Of Error 63

TABLE 5.7: PlR Before Using Fec And PLR After Using Fec 65

TABLE 5.8: The Mean Of PLR Before And After Using Fec. 66

TABLE 5.9: The Confidence Interval Of PLR Before Using Fec 67

TABLE 5.10: The Confidence Interval Of PLR After Using Fec 67

TABLE 5.11: The Result Of The Simulation Experiment 68

TABLE 5.12: The Result of the Simulation Experiment 70

List of Abbreviation

ARQ	Automatic Repeat Request
AWK	A Graphical Simulation Display Tool
C++	Object Oriented Programming Language
CBR	Constant Bit Rate
DT	Drop Tail Management
ECC	Error-Correcting Code
FEC	Forward Error Correction
ECN	Explicit Congestion Notification
FTP	File Transfer Protocol
IP	Internet Protocol
Ns-2	Network Simulator 2
OTcL	Object-oriented Tool Command Language
PLR	Packet Loss Ratio
QoS	Quality of Service
RNG	Random Number Generator
RTP	Real-time Transport Protocol
RTT	Round Trip Time
RS	Reed-Solomon
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol over Internet Protocol
UDP	User Datagram Protocol
UDP/IP	User Datagram Protocol over Internet Protocol

Chapter 1

Introduction

1.1 Introduction

This chapter provides an overview of the entire study. It contains a general overview for each section through this report structure. In section 1.2 we present the background of this study, in section 1.3 the problem statement, then in section 1.4 the research goals, followed by the objectives in section 1.5, which directly goes through the research questions in section 1.6, the significance and scope of the study in section 1.7 and 1.8 sequentially and the structure of the report in section 1.9, and finally in section 1.10 we end this chapter with a small summary.

1.2 Background

Recently new types of distributed applications have appeared, called distributed applications [1], because they run on different end systems. These distributed applications have a major deference for the old applications, which is the need for new types of *service requirements* that the older applications do not need. For these distributed applications the services that were provided by the network infrastructures like Internet, differ according to the kind of application, where the *service requirements* provided to elastic applications like e-mail are different from those *service requirements* provided to multimedia applications like streaming video as a real-time application.

The reason behind this difference [2], that elastic applications like e-mail can manage the delay of data arrival, and the variance in delay time, but at the same time elastic applications cannot tolerate data loss. On the other hand, multimedia applications such as video streaming are sensitive to end-to-end delay and the variance in delay time, but can tolerate a little bit of data loss, because the content of video streaming must be delivered as quickly as possible with no delay and must be delivered continuously. So *timing considerations* and *tolerance of data loss* have a high priority to be considered as an important parameter to provide quality of services (QoS) to the invoking application.

These days, the most popular and spread of these distributed applications types as mentioned in [3], are *multimedia applications*. Multimedia applications can be defined as a combination of *discrete media data* like text and *continuous media data* like video; it gives meaningful information only when it presented in time, and because it consists of a sequence of data it is called continuous media.

Since, in this research our interested focusing in one of the most important multimedia application, which is streaming video over Internet. We prefer to give more descriptions about the process in how this application transmitted over Internet. The need for these details is essential to get more depth about the problem that the video encounter during its journey from source to destination. At the end as we realize the problems, we can get that the most important one is data loss. To solve this problem, we have to describe more details again in the available solution to solve like this problem, with video transmission over Internet as a real-time multimedia application.

Figure 1.1 illustrates video streaming over the Internet, and as we outlined above, because of timing considerations, all components involved in video streaming must be able to present quality of service (QoS) [2], for the video application; the availability of such QoS enhances the video presentation.

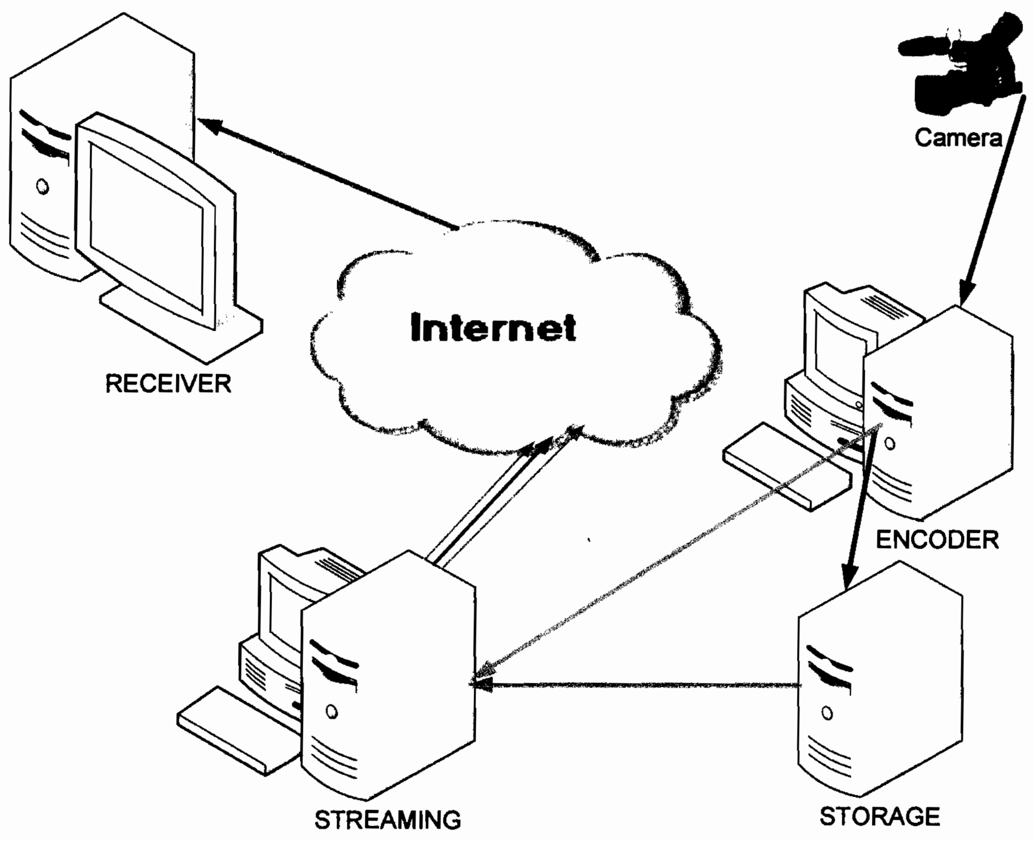

Figure 1.1: Video Streaming over Internet

According to [1, 2], streaming video as a real-time multimedia application requires quality of service (QoS) parameters, which are *bandwidth*, *delay*, and *loss requirements* to be effective. But the current best-effort network Internet does not offer any kind of quality of service (QoS) guarantees to the invoking application like streaming video.

From this viewpoint, and depending on the fact that through video transmission there must be a loss in data, researchers started looking for solutions for packet loss problem through video transmission, mainly to improve video quality. Researchers found that in the presence of packet loss, Application-Layer QoS control techniques provide good solutions; these techniques are employed by the end-system and do not require any QoS support from networks like the Internet. So by using this solution, the end-system-based recovery techniques should be used by the invoking applications like video applications, to achieve more quality of performance.

End-System-Based Recovery Techniques [4-6], are divided into two classes, sender-based techniques and receiver-based techniques to enhance transmission of streaming video applications in the presence of packet loss. Sender-based recovery techniques have two recovery approaches [29], Automatic Repeat Request (ARQ) and Forward Error Correction (FEC).

. One is Automatic Repeat Request (ARQ), this approach function is made by retransmission of the lost packet, and has been applied to the TCP (Transmission Control Protocol) for best-effort packet delivery; as a result of retransmission, ARQ will cause delays, which is not acceptable in streaming video because of its time constraints. As was mentioned in [7, 23], because ARQ schemes cannot be used to recover losses for real-time applications due to strict delay constraints, another approach needs to be used as an alternative for ARQ.

The second approach is Forward Error Correction (FEC), which would be a good solution to solve the problems faced by ARQ, because FEC has been suggested as a good mechanism for real-time applications like streaming video, to be used in the presence of packet loss. According to [24], the ability of FEC to recover from packet losses depends on the transmitted data that used FEC to solve the existing problem.

The goal of FEC is to recover from losses by adding [11-17], [25, 26] redundancy or increasing over the original data, which is redundant data called *parity*. This parity can be used to reconstruct the lost packets, or that packets arrive at their destination after the receiver's timing constraints. In fact this parity manages the receiver to correct lost packets with no need for using the ARQ approach, which means there is no need for the source to make retransmissions of the lost packets.

This duty of FEC is constrained by the amount of parity, which was sent and finally is used in case of packet losses. From this view, FEC contributes more toward improving QoS in the transmitting of video to achieve an acceptable and desired quality of presentation.

As was outlined above, the ability of FEC to recover from packet losses depends on the packet loss process. In spite of the benefits that we gain from parity, one major problem will occur, by adding redundancy the overhead would increase, which means that the amount of offered traffic on the Internet also increases, and with feedback congestion will increase, causing more packet loss. The packets lost are the losses in the FEC block, until finally the number of packets lost in the FEC block has a correlation with FEC performance.

This what [24] included in his work, and proved that there is a correlation between the FEC parity size and variance in packet loss, which leads to different level of FEC performance that finally will introduce different levels of presentation quality. Either experimentally it proved in [27, 28], that different FEC packet sizes introduced different levels of packet loss. From this view, and as a result of what we discussed above, FEC has to be introduced with minimum overhead for real-time applications, which leads to a search for convenient FEC parity amounts.

Before we make any more progress in this project, we will adopt the results that were found in [27, 28], in the first work the researchers found the suitable FEC packet size that guaranteed the best level of video presentation, the second work found the appropriate FEC block sizes that enhanced the video quality. With this project we continue the works of the researchers and enhance their works, by determining the appropriate FEC parity sizes that will guarantee high quality video streaming, depending on the results that were proposed on [27, 28], and by using network simulator 2 (Ns-2) to investigate and evaluate the appropriate FEC parity amounts.

1.3 Problem Statement

Over the packet-based Internet, in transmitting real-time interactive applications like streaming video [14], it uses unreliable protocols like UDP, and as a result of that, high packet loss rates would occur, because packet losses are unavoidable when we use protocols like this. Packets are considered lost if they are dropped due to congestion at routers, or they may reach the destination too late to be considered useless or lost, where these losses could actually be prevented. As a result of one packet loss [29], the received data will be not complete, so it would appear as degradation on the screen for video, thus the presentation quality would decrease, which is something that makes the user unsatisfied.

Forward error corrections (FEC) recover packet losses in multimedia applications such as real-time applications, and are recommended to improve video streaming quality in the presence of packet loss [7-9]. Because the heterogeneity of the FEC packet size, delay and loss and loss ratio will vary [30], this causes different levels of FEC performance. In [19] the researcher found that even with all values of lost packets that came as a result of FEC overhead, FEC was still the most appropriate mechanism that could be used to improve video streaming quality in the presence of packet loss.

In [31] the researcher showed that even with the FEC mechanism many problems still needed to be solved and required more efforts with the fact that different FEC redundant sizes caused variations in packet loss, which would have an effect on the quality of video presentation, and the researcher found with experiments, that the coding/decoding delay increased with the block size, and found with quantitative assessment that FEC coding increased the packet-loss rate and increased the delay, as a result of using FEC in improving video transmission. This result was the same as [5] showed in his research, where delay introduced at the receiver depended on the data generation time of a block of packets at the source.

As was outlined above, many researchers improved the utility of using the FEC approach as a suitable mechanism to recover from packet loss for transmitting video over Internet. But the main problem has to be solved, in that adding redundant packets into the network will increase the packet-loss and packet-loss ratio, which will affect the end-to-end video quality. From this point there is a need for investigating and evaluating the suitable FEC redundant amounts that would be sent from the source to the destination.

1.4 Goals of the Study

The project goals must be developed to serve as a direction, towards which the project is moving. At the outset, the goal would be based on the following question:

- To find the appropriate FEC redundant amount that optimally supports video streaming.

The goal would be to answer the following question:

- The optimal FEC redundant amount to be sent from server to client, which gives a lower level of packet loss, to achieve a high range of reconstructed video quality.

The answer to this question will shape the specific list of objectives that are crucial to achieve video streaming with high quality in the presence of packet loss.

1.5 Objectives of the Study

a. To identify the suitable FEC redundant amount that will be sent from the source to the destination to accomplish perfect compensation in the case of packet loss, by using Ns2.

b. To evaluate the FEC redundant amount found in the first objective, in order to verify the optimal FEC redundant amount.

1.6 Research Questions

What is the most appropriate FEC redundant amount to be sent from server to destination in the presence of packet loss, that is capable of supporting transmission of video streaming?

1.7 Significance of the Study

The current best-effort Internet does not offer any quality of service (QoS) guarantees to any invoking application like real-time applications such as streaming video. Hence a scheme is needed to transmit data over long distances with unreliable UDP protocol over the Internet, where there is no retransmission for lost data.

Forward error correction (FEC) provides a promising solution to the problem, where those errors are corrected at the receivers without the need to wait for the retransmission of packets. According to [10, 11], with the FEC solution the highest priority is to choose an FEC scheme that can achieve the desired loss recovery, which is to find the suitable FEC redundant amount that guarantees the least amount of overhead. Actually looking for the optimal parity amount that is being transmitted, which in the end will guarantee the best level of presentation quality, will provide us with the most appropriate FEC method. Furthermore by finding this value we will have already achieved our objectives.

1.8 Scope of the Study

The scope of this study is primarily focused on finding the most optimal FEC parity amount, to improve video quality in the presence of packet loss. According to [23], Application-Layer QoS control techniques provide a good solution; these techniques are employed by the end-system and do not require any QoS support from the network.

The major mechanism that would be used is forward error correction (FEC). With the results emphasized in [10, 11], on the Internet, forward-error correction (FEC) is a prominent error correction mechanism that has been widely used because of its ability to reconstruct the received videos in the presence of packet loss. Furthermore we will adopt the results proposed in [27,28], where the two researchers found the suitable FEC packet and block size that guaranteed a high level of video presentation, and the suitable parity amount that would guarantee high quality presentation of videos.

1.9 Report Structure

During this project we present proposed solution for losses occurred during video transmission, by matching suitable FEC parity amounts that finally incorporate the suitable FEC block and packet size that was found in [27,28], which would guarantee high quality presentation for streaming video, through decreasing the amount of packet losses and the packet loss ratio. Depending on this, our thesis is organized as follows:

Chapter 2 introduces the previous studies of the content related to our study, and it provides important information about the whole project, specifically Forward Error Correction, and how it can manage in the presence of packet loss; advantages and disadvantages of FEC; major problems with FEC and FEC parity amounts; the mechanism of FEC, and error-correcting code (ECC); and some information about the video streaming delivery mechanism; and major problems in video streaming, the bandwidth reservation, end-to-end delay, and reasons for packet loss, and how video streaming is affected by packet loss, then the proposed solution in Section, ending this chapter with a comprehensive summary.

Chapter 3 introduces a description of the methodology used in this study, in order to achieve the proposed objectives that have been previously discussed in section 1.5. Moreover, Chapter 3, which relies on the waterfall model, has seven major stages that the research passed through in order to produce the final approved solution for the problem defined. This chapter starts with an introduction, followed by a definition for the Network simulator, then we define experimental objectives, then move to the second phase to define the simulation model and environment, then with phase three we define the performance metrics, followed by the simulation phase as stage 4, then stage five describes the simulation process, with phase six being the results of the analysis and data interpretation, and the last phase is a presentation of the results, with a summary ending the chapter.

In Chapter 4 we go through phase four and phase five of our research methodology, where the simulation model, performance metrics and environment (scenario, topology, and configuration) are put together using a network simulator as input script and run the simulation 20 times for each parity amount. In stage five of the research, data collected from simulation runs in the previous section are input, where the output for this experiment is the trace file. The trace file consists of twelve columns, each representing an individual event taking place during the simulation phase.

In Chapter 5 we present phase six and phase seven of our research methodology, phase six presents the Trace-file analyses, where we discuss, evaluate, and interpret the results gained. Then we used the cropped data as an input for a graphical simulation display tool called an *awk*. And through that tool we can see our data presented in a graphical mode. In stage seven we present the final phase of the research methodology; this is important because it is the final stage that our methodology has to move into, and this phase involved presenting and showing the results from the simulation.

In Chapter 6 which represent the end of this study, this chapter is very important to the entire body of the research, because it is the final stage that the project has to move in, it is involved presenting and showing the results from simulation. Depending upon the analysis of cropped results, our conclusions and the contribution of this project introduced in this chapter

1.10 Summary

In this chapter, a concise introduction has been presented about the proposed solution, which is to find the suitable FEC parity amount that guarantees the least level of packet loss in accordance with the desired objectives; it also offered a clear view of the scope and the significance of the study.

Chapter 2

Literature Review

2.1 Introduction

In this part of the project, we made a review of literature related to Forward Error Correction as an error control mechanism using with streaming video, and how it can manage in the presence of packet loss. During that we will mention information that makes our details more obvious.

At the beginning, section 2.2 starts with an overview about the video streaming delivery mechanism, and in section 2.3 the major problems facing streaming video through transmission over the Internet are introduced; next in section 2.4 we identify packet loss, directly followed by subsection 2.4.1 for the reason behind packet loss being introduced; in subsection 2.4.2 how video streaming can be affected by packet loss; in subsection 2.4.3 how to recover lost packets; the main reason behind choosing forward error correction in the presence of packet loss for video streaming will be in section 2.5, followed by the influence of using FEC as an error correction mechanism to overcome loss; listing the advantage and disadvantage of using FEC in subsection 2.5.1, which leads to the next step, major problems with FEC and FEC parity amounts in subsection 2.5.2; then we explain how FEC works in subsection 2.5.3 as a mechanism of FEC, and subsection 2.5.4 introduces the error-correcting code, and finally in section 2.6 the proposed solution is presented, and the chapter ends with a summary.

2.2 Video Streaming Delivery Mechanism

Using elastic applications with multimedia applications together in one form of data by using computing power, allows us to represent information interactively in the computer. And then we can transmit this data over a computer network like transmitting video over the Internet. We can present and broadcast this video in two ways [29], by *a live* or a *recorded* fashion, broadcast can be either *analog* or *digital*, digital on-line multimedia may be *downloaded* or *streamed;* streaming multimedia may be *live* or *on-demand.*

The major difference between the download method and the streaming method is that the user [2], with the first mode, downloads the whole video content and then runs the video. In the second mode, the user doesn't need to download the whole video content, instead the user can play the video data, while at the same time is still receiving the other parts of the data.

As was maintained in [22], because streaming video consists of a sequence of media data, it became known as continuous media, where if this video is presented at the client-side in time, it gives a good presentation. The basic mechanism of sending and receiving video streaming is that video is sent in compressed form; by this way we decrease the required bandwidth to transmit the video.

We can summarize the mechanism of transmitting video over the Internet as follows [32], first at the sender-side, the video data is divided into packets, these packets are then encoded by using any normal compression techniques, then these packets are transmitted over the Internet to their destination using a transport layer, and secondly at the client-side the receiver can decode the received video packets and play it back. As we mentioned above, the user with streaming mode does not need to wait to deliver the whole content of data, by using this way the user can save time. And in order to prevent variance in the arrival time of packets, usually at the client-side there is a short delay between the beginning of playback and delivering the video data of about 5-15 seconds; by using this delay we can extend the quality of video presentation.

2.3 Major Problem in Video Streaming

Real-time multimedia application like streaming video, needs quality of service (QoS) [1, 2], in order to be effective and function well. These QoS parameters are *bandwidth, delay, and loss.*

The fact is, current best-effort Internet does not offer any quality of service guarantees to the invoking streaming video application. Streaming uses either TCP (Transmission Control Protocol) or UDP (User Datagram Protocol) as the transport layer, where both run over IP (Internet Protocol), it follows that neither of these transport protocols make any quality of service guarantees for the invoking application.

In fact, the Internet moves data from the sender to the receiver as quickly as possible, as it's best-offer to the invoking application, and at the same time the Internet makes no guarantee about bandwidth, loss and delay that may happen for the video through its journey from source to destination over the network.

2.3.1 Bandwidth Reservation

In order to achieve acceptable presentation quality, [25] indicated that minimum bandwidth requirement is one of the most important QoS requirement parameters for real-time video applications. But the Internet does not provide bandwidth reservation to meet such a requirement. The main idea here is to looking for a solution in the absence of Internet aid to solve the bandwidth problem.

The bandwidth problem can be seen from two points [33]; if we transmit the video faster than the available Internet bandwidth, in this case congestion would occur as a result of this increase in traffic, which will lead to packet loss. And in case we transmit the video slower than the available Internet bandwidth, we have a problem at the receiver-side where the presentation quality will decrease. So the need for matching the transmitting of our video to the available Internet bandwidth arises to prevent these problems.

Furthermore, to solve this problem, and with the fact that the Internet has limited transmission capacity, the importance of the compression mechanism arises [26], which helps us to minimize the size of the sending data of the video at the

2.3 Major Problem in Video Streaming

Real-time multimedia application like streaming video, needs quality of service (QoS) [1, 2], in order to be effective and function well. These QoS parameters are *bandwidth, delay, and loss.*

The fact is, current best-effort Internet does not offer any quality of service guarantees to the invoking streaming video application. Streaming uses either TCP (Transmission Control Protocol) or UDP (User Datagram Protocol) as the transport layer, where both run over IP (Internet Protocol), it follows that neither of these transport protocols make any quality of service guarantees for the invoking application.

In fact, the Internet moves data from the sender to the receiver as quickly as possible, as it's best-offer to the invoking application, and at the same time the Internet makes no guarantee about bandwidth, loss and delay that may happen for the video through its journey from source to destination over the network.

2.3.1 Bandwidth Reservation

In order to achieve acceptable presentation quality, [25] indicated that minimum bandwidth requirement is one of the most important QoS requirement parameters for real-time video applications. But the Internet does not provide bandwidth reservation to meet such a requirement. The main idea here is to looking for a solution in the absence of Internet aid to solve the bandwidth problem.

The bandwidth problem can be seen from two points [33]; if we transmit the video faster than the available Internet bandwidth, in this case congestion would occur as a result of this increase in traffic, which will lead to packet loss. And in case we transmit the video slower than the available Internet bandwidth, we have a problem at the receiver-side where the presentation quality will decrease. So the need for matching the transmitting of our video to the available Internet bandwidth arises to prevent these problems.

Furthermore, to solve this problem, and with the fact that the Internet has limited transmission capacity, the importance of the compression mechanism arises [26], which helps us to minimize the size of the sending data of the video at the sender-side before the transmitting phase, which has good benefits to solve a part of the bandwidth problem.

2.3.2 End-to-End Delay

Each single video packet must arrive at the destination in a specific limited time to be decoded and displayed because real-time video as an interactive application must be played out *continuously* [34]. For this reason video requires bounded end-to-end delay.

As a result, and because the Internet also does not make any promises about any type of packet delay within packet stream, we can notice two types of delay with the transmitted video - the packets that violate the receiver constraint time and the variation in time arrival of packets for the same data; this variation is referred to as the delay jitter. Delay jitter is a problem because the receiver must receive/decode/display packets at a constant rate. And any late packets resulting from the delay jitter can produce problems in the restructured video, which affects finally the quality of the presentation at the client-side, which makes them unsatisfied.

For solving this problem, at the client-side users have a small waiting period between starting of the playback and the delivery time (about 5-15 seconds); actually this delay is very essential in video streaming and provides huge benefits, because it reduces the effect of variation of packet arrival times.

2.3.3 Packet Loss

Continuous media applications like real-time video streaming require data to be delivered within a specified time, but with the fact that real-time video streaming applications use UDP over the Internet, it causes higher traffic on the Internet [23]. And as a result of the increasing in traffic, congestion would definitely occur, so packet loss and delay could be very high during network congestion, which will cause a decreasing of video quality levels.

Actually, Internet does not make any promises about the packet loss, furthermore, we can notice that for real-time applications we can say that our packet is lost in two cases; firstly, if the packet is dropped in the route to the destination, secondly, it is considered lost if the packet arrives at the destination after the receivers' time constraints.

Finally, any data lost in transmission definitely will not be used at the receiver-side. Now because of the absence of Internet aid in such a situation and because losses have destructive effect on the video quality, video streaming applications are designed with error control [34].

In this research we are going to investigate the third QoS parameter of real-time requirement, which is *packet loss*, and we will identify packet loss, reasons behind the loss, how video streaming is affected by packet loss, and how to mitigate loss. Then we will discuss mechanisms that are used to recover from packet loss; furthermore we will finally investigate and evaluate the suitable FEC redundant amount that would be sent from the source to the destination to accomplish perfect compensation in case of packet loss.

2.4 Video Streaming and Packet Loss

Suppose the case where we have two end-systems communicating over a network such as the Internet; in this case data will travel between the two end-systems. This travelling data would be considered lost [34] if any part of this data was dropped in its route to its destination, or in the case of real-time applications, arrive too late to be used by the receiver. The problem of packet loss actually depends on the type of application, for example, for elastic applications it cannot tolerate data loss. On the other hand, multimedia applications such as video streaming can tolerate a little bit of data loss. Where we have to take into consideration, in transmitting real-time applications over the Internet like video applications, is that there will be packet loss through its journey from source to destination.

Since our project focuses on real-time video streaming using UDP over the Internet, congestion problems will occur as we mentioned in section 2.3.3; this will cause unwanted delays and loss of data, hence we can give another definition for loss [33]. With video streaming, packets are considered lost in two cases; firstly, if packets are dropped through the route to destination or if packets are not available at the receiver-side at the required playout time.

2.4.1 Reason behind Packet Loss

Transmitting video over the Internet will face packet loss [34-37], for several reasons and caused by the high traffic; this high traffic comes as a result of huge amounts of data delivered on the Internet. If these delivered data exceeded the capacity of the Internet, congestion will occur and this congestion will cause packet loss, either dropped and never reached the destination, or was too late and became useless to be used in the video construction at the receiver-side, because it violated time constraints.

As we outlined above, real-time applications can tolerate a little bit of data loss; what will happen in the case of increasing losses for real-time video streaming, which comes from network congestion; compared to the limited tolerance for loss, this increase will cause a decrease in the level of the quality of the video at the receiver, and the next section will cover this problem.

2.4.2 How Video Streaming is Affected by Packet Loss

As we outlined above the increasing in packet loss comes as a result of higher traffic; in the case that we have limited network bandwidth, this will cause congestion to occur. When congestion occurs, packet loss will take place as a result of this congestion.

Streaming video is bounded by two main principles; to be presented as well as the users want, [38] the receiver has to accept the entire amount of video packets, and the receiver must receive these packets within a time constraint, as the outcome of these two is what will happen to the video presentation if the loss takes place within one of its criteria (packets dropped or violate receiver time constraints).

In order to reconstruct the video at the receiver-side, the receiver has to have received all video packets that had been sent by the sender, otherwise, if the loss occurred for one packet either by being dropped or latency, this will cause degradation in the video quality and finally the client will be not satisfied [35].

2.4.3 How to Recover Lost Packets

The time limitations for the recovery of error are different according to which type of application we deal with, continuous or discrete media. So, a separate error recovery mechanism should be used accordingly for each media. However, for this project, since it deals with transmitting continuous media over a best-effort network, in other words, we deal with transmitting video over Internet. We can mitigate the effect of losses by using error control techniques.

High traffic for the Internet comes as a result of the popularity of the Internet, where we can notice an increasing demand of delivering video over unreliable channels like Internet [34], which finally will increase congestion; more congestion causes more delay and unwanted packet losses. Because of these reasons the need for error control in transmitting video over the Internet becomes an important issue.

Error control is a mechanism that needs to maximize video presentation quality, in the presence of packet loss, where it seems that up-to-date transmitting of real-time video over the Internet has to have some packet loss [39, 40]. For that reason video streaming systems are designed with error control to be able to combat the effect of losses [35]. Forward error correction (FEC) is the mechanism that will be used in this case, because of its advantages over other mechanisms like Automatic Repeat Request (ARQ) mechanisms.

2.5 Forward Error Correction (FEC)

Mainly to improve video quality, we have to consider the effect of losses and delays that the packets of video suffer during their journey from the sender-side to the receiver-side.

Application-Layer QoS control techniques provide a good solution; these techniques are employed by the end-system and do not require any QoS support from the network. Sender-based recovery techniques have two recovery schemes to enhance transmission of streaming video application in the presence of packet loss, Automatic Repeat Request (ARQ), and Forward Error Correction (FEC).

The first one is Automatic Repeat Request (ARQ), this approach functions by retransmission of the lost packets, and has been applied to the TCP (Transmission

Control Protocol) for best-effort packet delivery; as a result of this it will cause delay, which is not acceptable in real-time streaming because of time constraints. According to [2, 4, 5], because ARQ schemes cannot be used to recover losses for real-time applications, due to its strict delay constraints, another approach needs to be used to be an alternative for ARQ [4-6], [25, 26].

Forward Error Correction (FEC) would be the perfect solution to solve this problem. FEC has been more commonly suggested for real-time applications, according to [29]; the ability of FEC to recover from packet losses depends on the packet loss process. Here we start to talk about other factors, what the losses process depends on.

In this project we will focus on the primary factors related to and that affect FEC performance, [41, 42], *FEC Block size, FEC Packet size*, and *FEC Redundancy*. The goal of FEC is to recover from losses by adding redundancy (The addition of redundant packets is called (parity)), which can be used to reconstruct any lost packet or packet that violates the timing constrains at the receiver. According to [7-23], [43-47] all of those researchers agreed in their research, that in the presence of packet loss for real-time video streaming, it was obvious that FEC had the ability to recover this loss.

Now to guarantee a high level of FEC performance [27, 28], we have to investigate the suitable FEC Block Size, FEC Packet Size, and FEC Redundancy, and in order to avoid the main disadvantages of FEC, which is that all approaches of FEC design to reconstruct the predetermined lost amount of data, so even with no loss FEC will still have overhead, is to find the proper control of FEC overhead, which can definitely provide the quality of received video over lossy channels.

2.5.1 Advantages and Disadvantages of FEC

As outlined above, retransmission and the rate control mechanisms are often considered unacceptable for interactive real-time video applications, because they increase end-to-end delay, which is against timing constraints, and another reason against using retransmission is that retransmission may increase congestion.

According to [31, 32] the main reason behind choosing FEC to enhance the performance of real-time video applications in the presence of packet loss is primarily because of its advantages compared to retransmissions; the FEC approach produces a small transmission delay because the FEC mechanism is not constrained by Round-Trip-Time (RTT) and there is no need for a back-channel, and there is a major reason behind using FEC to enhance video streaming, which is that video is considered as an unreliable multicast application because -as was maintained earlier- it can tolerate some errors, and because of that video streaming systems are designed with error control to enhance presentation quality in the presence of packet losses.

In fact, and as it was shown in [36], using forward error correction for error control has been an acceptable solution But, with all of those benefits FEC-based approaches still have to face the principal problem, which is considered an important point, that all approaches are designed to reconstruct the predetermined lost amount of data, so even with no loss FEC would still have overhead.

2.5.2 Major Problems with FEC and FEC Parity Amount

To determine if the received data is the same as the original sending data is the main challenge for the receiver, where the ability of the receiver to detect and correct errors depends on the chosen error control mechanism that was used. Forward Error Correction (FEC) is a suitable mechanism for error control for transmitting video over the Internet; it is a method of transmitting data without loss, which is used in the presence of packet loss, by enabling the receiver to detect and correct errors in the received data.

The goal of FEC is to add redundancy that can be used to recover from packet loss. But the main problem with FEC is that loss characteristics for packet networks are often unknown and time varying. So, [22] FEC may take three characteristics: *effective* (if the redundant data is sufficient for the lost data), *ineffective* (if the redundant data is less than the lost data), *or inefficient* (if the redundant data is more than the lost data). Our research focuses on this problem and looked for a solution by matching the appropriate FEC parity amounts with FEC blocks that can achieve effectiveness, to improve aggregate FEC performance when used by the source in the network. Because if there is enough redundant data included in the stream, the corrupted data can be recovered.

FEC can achieve this by sending redundant data with the original data in order to reconstruct the received video; this achievement has the primary problem of the additional bandwidth requirements, which happens because of the parity amount added to the original data. In the literature [38, 39], the performance of FEC depended on several factors, the major factors are the FEC block size, the FEC packet size and parity amount, the size of the FEC block, and the size of the FEC packet is related to the packet loss and packet loss ratio that the received packets face during their journey from the sender-side to the receiver-side, as mentioned in section 2.4, the amount of loss has a correlation with FEC performance, which finally has a correlation with video quality presentation.

2.5.3 FEC Mechanism

FEC has two levels, byte-level FEC and packet-level FEC, and according to this the symbol within the codeword is taken so that the first level symbol will be byte and the second level symbol will be packet. In our project we are interested in the second level, which is the packet-level FEC.

The goal of FEC is to add (extra) redundancy to a compressed video, and we refer to these redundant packets as parity packets. In this technique redundant packets are generated from original data packets based on a linear block code algorithm; the redundant data is transmitted with the original packets, so when loss occurs in the original data at the receiver-side, the redundant packets can be used to recover from packet loss. The block codes can be used only for loss recovery because the loss location is easily identified using the sequence numbers of real-time transport protocol (RTP) packets.

According to [48], if we have (k) original data packets encoded into (n) packets, the total amount becomes (n), and (n-k) is the number of redundant packets being used for loss recovery. So based on any received (k) packets out of (n) encoded packets, the receiver can recover (k) original data packets.

2.5.4 Error-correcting code (ECC)

The technique for error-correction coding (ECC) enables the decoder at the receiver-side to correct lost packets without requesting retransmission of the original data; for this reason this coding technique has been used to improve the reliability of transmitted data.

To correct errors that occur during video transmission, we can use different types of error correcting codes (ECC), where the basic principle [49] for all types are the same; redundant data is added to the original data in order to correct errors that may occur in the received video. Practically redundant packets are appended to data packets, to obtain a coded sequence or a code word. Figure 1.3 shows block encoding for error correction. For simplicity, we can say that the input for the encoder at the sender-side is a digital data sequence which the source sends to the encoder. The encoder adds extra data (or *parity*) aside from the original data, so we can guess that the output would be a longer sequence of code, called a codeword. After the codeword has been transmitted to the receiver-side, which has the same type of decoder, the original data sequence is extracted in order to reconstruct the video. Figure 2.2 shows a digital communications system.

An *error correcting code* will consist of the sum of all code sequences or code words. As we precede we can use one of two types of ECC, the most popular used these days with video transmission as ECC is block codes, for linear block codes, because the output block encoder (code words) are independent from each other; in other words the output does not depend on the previous output, so the process of block codes will be independently block by block.

Sender-side and receiver-side must include the same coding scheme that transmitted videos have [50]. At the sender-side the ECC encoder takes the data from the source as input and adds redundant data to it, so that most of the errors can be corrected. At the receiver-side, the ECC decoder uses the redundant data with the original data in order to reconstruct the received video, at the same time correcting errors that take place at the receiver-side. Actually we have several types of block codes that we can use to accomplish error correction in the presence of packet loss; for our project we will use Reed-Solomon (RS) codes, which are the most used in video transmission.

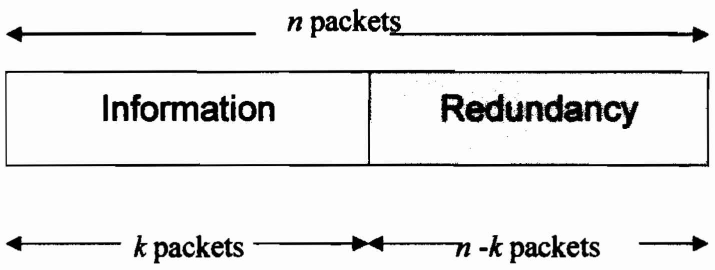

Figure 2.1: Block encoding for error correction

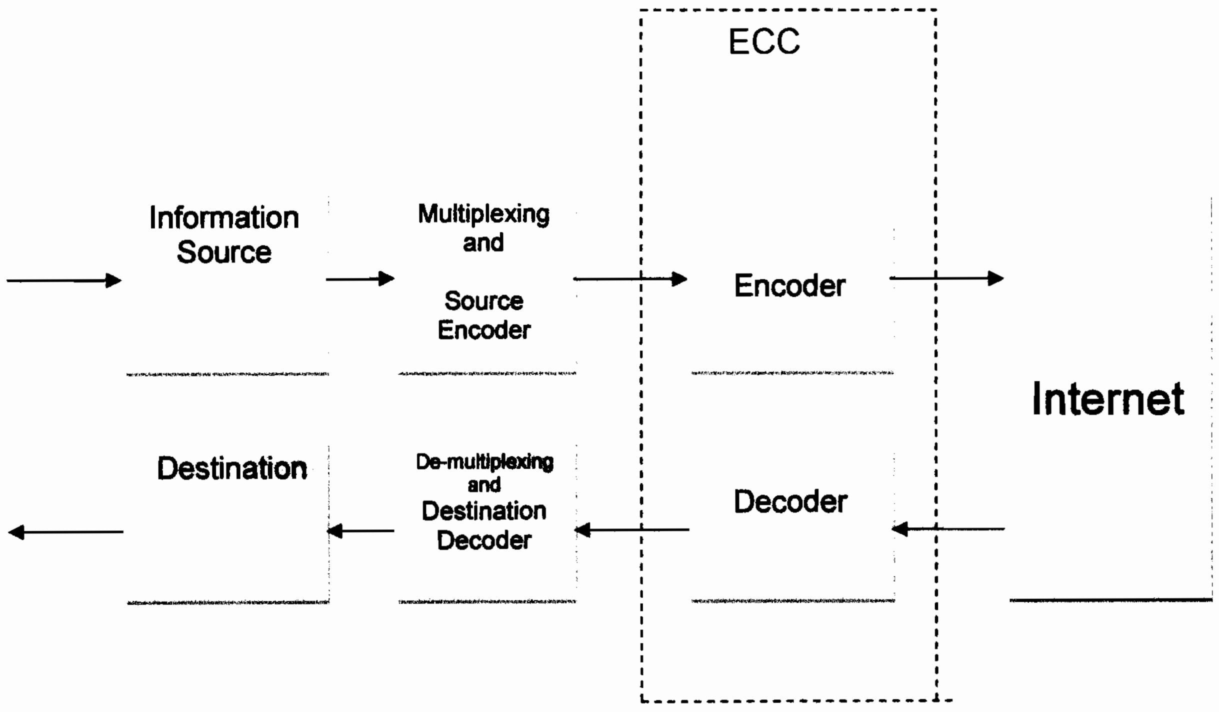
ECC
Information Source
Multiplexing and Source Encoder
Encoder
Internet
Destination
De-multiplexing and Destination Decoder
Decoder

Figure 2.2: Digital communications system

For transmission of data over networks like the Internet, we have several Error-Correcting Codes. The most prominent one is the linear block code that is called the Reed-Solomon (RS) code. With Reed-Solomon codes, we put a block of data, which is the original data as an input. At the other side of an RS encoder we obtain a codeword as an output; this output contains the original data and redundant data. So, in this case the sender sends the data as encoded blocks, and the number of packets (*n*) in the encoded block is:

$$n = 2^m - 1 \tag{2-1}$$

A Reed–Solomon symbol size of eight bits:

$$n = 28 - 1 = 255 \text{ packets per block}$$

For that, the standard (255) Reed–Solomon code is capable of correcting up to 16 Reed–Solomon symbol errors in each codeword.

A Reed–Solomon code is specified as RS (*n*, *k*) [51,52], here the encoder adds redundant information to the stream of input packets using a particular algebraic algorithm; for this reason there must be more packets at the output of the encoder than at the input, so that n > k.

The decoder at the receiver-side exploits the redundancy to correct any errors that may have been happened, where the decoder uses the inverse of the algebraic algorithm that was used at the encoder to identify and correct any errors.

The ability of the Reed-Solomon code to correct errors is determined by ($n - k$), which is the parity amount in the block. Now this ability is limited by the main factor, which refers to the location of the error. If error locations are known, in this case Reed–Solomon code can correct up to ($n - k$) error packets, otherwise Reed–Solomon code can correct up to ($n - k$) / 2 error packets.

In our case since we investigate video transmitting, RS can be used to overcome the unreliable nature of data transmission over the erasure channel, the encoding process assumes a code of RS (*n*, *k*). If we represent (*t*) as the number of packets that contain errors in a codeword, then:

$$t = (n\text{-}k)/2 \tag{2-2}$$

Data packets in the block are a design parameter, so it can vary according to the used size of FEC block and we can get from the previous equation (2) that:

$$K = n - 2t \tag{2-3}$$

2.6 Proposed Solution

The large body of this research work will be based on [27, 28], the results of the experiments that were proposed in the researches enhanced those facts that we mentioned in section 2.5.4, by confirming that the size of the FEC block, and the size of the FEC packet has a relation with the packet loss and packet loss ratio.

The amount of loss has a correlation with FEC performance, which finally has a correlation with video quality presentation. The researchers found from experimental results that the more suitable mechanism that we can use in the presence of packet loss during video transmission is FEC, and to guarantee a high performance level for FEC, we have to match a suitable FEC block size and an FEC packet size that guarantees a high level of FEC performance in [27, 28], (200 Packets, 1100 Bytes) for FEC block size and FEC packet size, respectively. From this view we will continue our research to find the suitable level of redundancy according to these results, as a part of the solution to guarantee perfect FEC performance, which will achieve loss recovery while minimizing computational overhead.

2.7 Summary

After presenting this chapter, it is clear that using the FEC technique as an error control mechanism to recover from losses has its advantages over the use of other mechanisms; especially when it used in transmitting real-time applications over the Internet. But in spite of these advantages still some disadvantages have to be solved, in order to increase the utilization of this mechanism, and to improve its performance. The next chapter will discuss the methodology of the research.

Chapter 3

Research Methodology

3.1 Introduction

Previously, in Chapter Two, discussions were focused on the literature of the previous studies of the forward error correction based research. In this chapter discussion targets the research methodology used in this study; this study will adopt the general methodology of design research and was structured with seven phases, known as the life cycle of this research [47]. This life cycle was established to investigate and evaluate the chosen parity amount that would be added at the encoder, and is known as the waterfall model or methodology because the result of each phase flows into the next phase.

We start this chapter in section 3.2 by introducing a brief description of the need for simulation with a description of network simulator 2 (ns-2). The methodology starts with the definition of experimental objectives in section 3.3, then moves to section 3.4 where the second phase will define the simulation model and environment, in section 3.5 phase three will define the performance metrics, followed by the simulation phase in section 3.6, then in section 3.7 stage five of simulation process; next in section 3.8 are the results of the analysis and data interpretation, the last phase will be the presentation of results in section 3.9. Figure 3.1 illustrates these phases graphically.

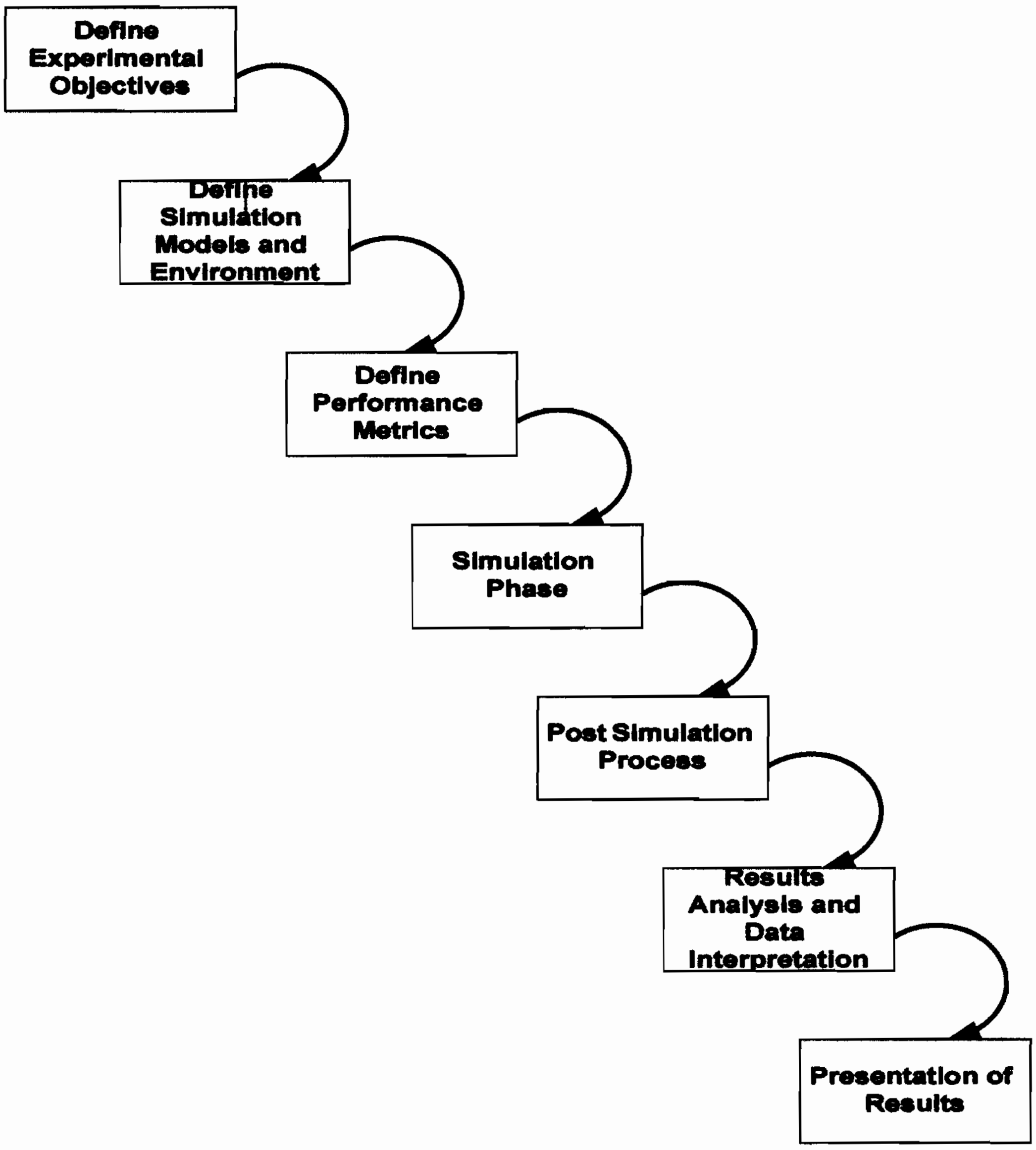
Define Experimental Objectives
Define Simulation Models and Environment
Define Performance Metrics
Simulation Phase
Post Simulation Process
Results Analysis and Data Interpretation
Presentation of Results

Figure 3.1: Simulation phases

3.2 Network simulator

We were interested in this project to evaluate the network performance during the process of transmitting video from source to destination using FEC as the error control. There are several approaches to conduct performance evaluation of a communication network, for our research we use the simulation technique to evaluate the performance of FEC [53], the reason behind this choice was because of the real power of simulation, which is the ability to represent the behavior of complex networks and protocols, and it allows for efficient investigations, by using numerous parameters and then analyzing action sequences efficiently and quickly on the network and protocol design. Where at the same time simulation provides an environment that is effective for training.

For these reasons in networking system, most researchers used simulation as their main research methodology. According to [54-56], with real-time simulation researcher can perform analyses using real-time system parameters; at the same time researcher can also simulate scenarios, in this way they can modify various attributes, to observe how the network would behave under different conditions, in order to observe end-to-end performance, so network simulation, which is actually a software program that can models the behavior of the network, and after we design the model, it will be executed and analyzed. Most of the protocols in use these days, like TCP, UDP, and IPv4, are supported by most types of simulators.

For recent research [53], the most prominently-used network simulator is ns-2, which can run on Widows and Linux and Mac systems. Ns-2 is a powerful framework that allows the researcher to implement both the network topologies and any proposed protocols. At the same time ns-2 is an event-driven simulation tool, which means that a group of events is stored, because event-driven simulation proceeds in order from one event to another. With ns-2 simulation we can see that it is written with two programming languages, C++ and OTcL, where the C++ and the OTcl are linked together.

In the OTcl domain, where it does not contain any functionality, it actually acts as an interface that interacts with users of the simulator, and by using this interface researchers can make the desired installation of their input parameters and events, and through the same interface researchers can get their output data, produced as a trace-file. On the other side the functionality is defined in C++ objects, where the chosen event by the researchers takes place using this functionality.

So we used Ns-2 in our research to achieve the main goals of using simulation, which are firstly to recognize what we are going to build, secondly to make suitable scenarios for our model, thirdly to construct our topology, insert our parameters and events, then run the simulator and observe how the network performs under various conditions. Furthermore, to describe how our network will be, we used two main inputs for the simulator, the *parameters* and the *events*, the output being either *text-based* (trace-file) or *animation-based* simulation results, which show events that occurred in the simulation during the specified time; after that we used the trace files to analyze a particular behavior of the network, as an analyzed phase, by using (awk). We choose this methodology in order to execute our project, because it is the most appropriate way to investigate and evaluate the proposed solution that could extend the performance level of FEC. We start to activate this methodology in the next section until the end of the chapter.

3.3 Experimental Objectives

Through the process of the research cycle shown in Figure 3.1, this section takes place as the most of the important parts of the research since it is very important to know the objectives for developing the solution for finding the suitable amount of FEC redundancy, and the need for such a demand arising from recognizing the dimensions of the problem presented previously in section 1.3, and using the FEC approach as a suitable mechanism to recover from packet loss for transmitting video over the Internet. The main problem has to be solved; that adding redundant packets into the network will increase the packet-loss and packet-loss ratio, which will affect the end-to end video quality. From this point the need for identifying and evaluating the suitable FEC redundant amount that will be sent from source to destination will appear.

From an understanding of the problem statement and our objectives, we can obtain our simulation experiment that we build using Ns2. In our experiment we examine the effect of different FEC redundant amounts against two variables, *packet loss*, and *packet loss ratio*. And from the previous discussion in Chapter two, we can see that the performance of FEC is affected by packet loss and packet loss ratio, which finally affects the quality of video presentation. Now to have loss occur in our experiment during the transmission of video over the designed network in the simulator, we must activate congestion. We can do that by increasing the number of transmitting node, so we start with three senders where all transmitted data meet at one router, where the congestion is supposed to appear, followed by packet loss. When the loss occurs we can change our parity amount that would be 5, 10, 15, 20, 25, 30 and 35 packets, respectively, and from the results of the simulation output, we can collect our data. The collected data would be analysed and rationalized to find the suitable parity amount that achieves our goals.

3.4 Simulation Models and Environment

In order to make our model closer to the real world, we matched our research requirements to the real internet environment; we based it on the procedure used in [27, 28] to build up our simulation model. Through this work we can find the suitable FEC redundant amount that causes the least amount of packet loss and the lowest packet loss ratio, to guarantee the highest level of FEC performance that provides high quality for video presentation.

3.4.1 Simulation topology

In the previous section 3.3, we investigated our problem clearly, and described in general how our topology should be depending on our problem and goals. With this topology different FEC redundant packet sizes were examined, so simplicity and representative were the most desired requirements in such a case, which meant that we must design a simple topology that represent our problem and goals [41], Nodes and links are the fundamental elements in the network topology; nodes are linked up using links, the function of the nodes is to direct the incoming packets to the correct outgoing link. To achieve simplicity we used a single-bottleneck topology in our experiment. Figure 3.2 shows the simulation topology, having been used by other researchers [27, 28].

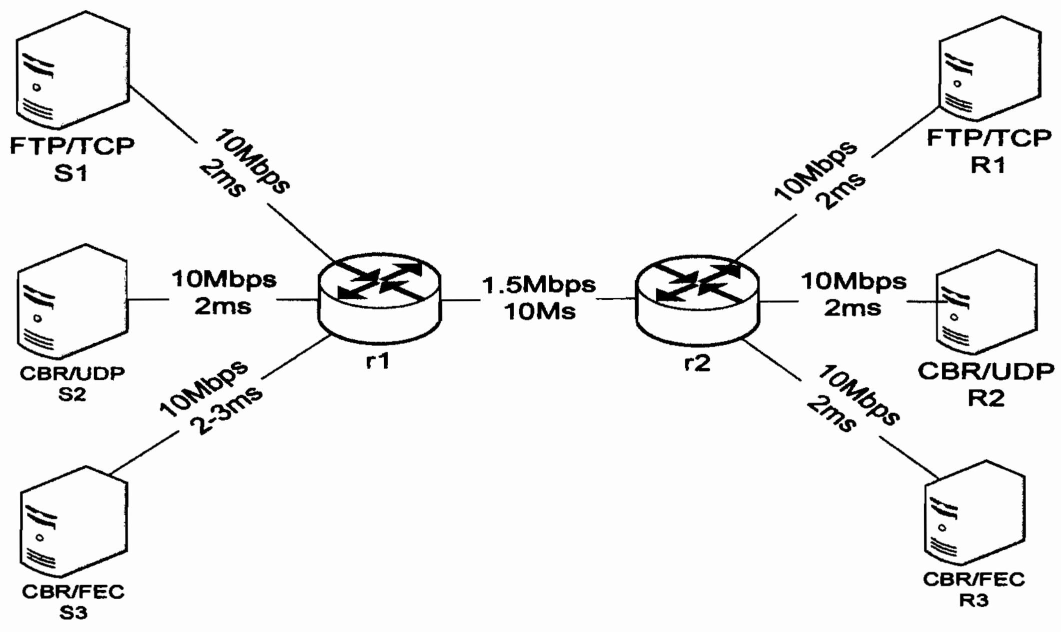

Figure 3.2: Simulation Topology

3.4.2 Traffic characteristics

We generated two sets of competing traffic patterns, FTP and CBR; then attached FTP to the TCP source, and CBR to the UDP source, and then CBR to FEC over the UDP source. As we noticed in our topology, we try to generate high traffic to affect the behavior of the link's traffic, where the amount of data flows are competing for network resources.

We see in our topology the bottleneck link is shared between three competing data flows of three senders and three receivers, which finally will influence the behavior of the network traffic. The main idea behind this structure is to create competing traffic environments that can generate congestion, to cause packet loss. So by achieving this loss, we can start to use our proposed solution by adding several amounts of redundant data, where finally we can achieve our goals.

3.4.3 Simulation Parameters

In setting parameters for our topology, we tried to be closer to real life and to present the real Internet, since the output of the simulation depends on these parameters; we tried to choose these parameters carefully because they affect the performance of the chosen design. Furthermore we made the values of these parameters depend on those values used in [27, 28] the reason why we chose the same parameter values is because our work is an extension of research outputs, to achieve our goals.

Through these values we determined the values of packets for FTP and FEC, the bandwidth of each link, the delay through these links, management policy, queue limits, FEC block sizes, FEC packet sizes, and FEC parity amounts.

3.4.3.1 Queuing Management Policy

In this research we used the Drop-Tail queuing management policy, where it is one of several types of queuing policies available with NS2 [53]. In Ns2 the queuing management represented as an algorithm; this algorithm handles the length of the packets queue. When this length is exceeded by the incoming packets, the algorithms determine when dropping packets has became suitable; when the buffer of the router has become full.

Now to determine which packet will be dropped first is the management policy we were talking about; the drop may occur with the last packet to arrive, or determined by limited priorities of the using policy, or drops could occur randomly. Furthermore, in our case we can use more than one type of these policies; for our research we use Drop tail or tail-drop (DT); by using this policy the last packet arriving that finds the buffer full would be dropped.

3.4.3.2 FEC Setting

As we outlined above, the performance of FEC depends on the process we use with FEC, the nature of the data that we use with FEC to send it over the Internet, these things actually lead into essential performance parameters for FEC [57], which are FEC block size, FEC packet size and FEC parity amount. These parameters we notice that there is an effect on the packet loss rate.

For this experiment we set the FEC block size and FEC packet size according to the experiment results in [25,26], because we adopted the two research results to make extensions using our proposed parity amount, that we hope it increases the performance level of FEC, so with this project we used various values for the FEC parity amount.

3.4.3.3 Bottleneck Bandwidth

We tried to choose the bottleneck bandwidth closer to the actual Internet. Bottleneck bandwidth is defined as the link (between r1 and r2) bandwidth or throughput, which can actually be estimated in the absence of any other traffic in the same link [58]; in other words it is the throughput that the link (between r1 and r2) can provide from source to destination where there is no other source sending data at the same time.

3.4.3.4 Bandwidth

As we outlined in section 2.3.1, that for transmitting video over Internet to be acceptable it needs to minimize bandwidth, and we show the influence in case the transmission of video is faster or slower than the available link bandwidth. During this research we were looking at raising the level of video presentation, and with our unguaranteed bandwidth from Internet and furthermore using UDP as the transport protocol, all these factors encouraged us during building our scenario to be closer to real life using bandwidth, to give our results more credibility.

3.4.3.5 Delay

Section 2.3.2 showed that during streaming video the delay between source and destination is very bounded, any packet violating the time constraints and arriving at the destination after that time is considered a lost packet, because the receiver cannot use this packet within the specified time to reconstruct the received video, so the packet is useless.

For that, real-time applications like video are considered very sensitive if the arrival of packets violated the receiver time constraints. Through this project we employed all facilities to prevent this delay, furthermore, one of the biggest reasons behind delay is congestion, so by achieving our research goals we have already aided in preventing such delays.

3.4.3.6 Simulation Data

For source traffic data we used several types to be more realistic, FTP and CBR. FTP (file transfer protocol) was used with TCP/IP, and CBR (constant bit rate) was used with UDP and with FEC over UDP, which represents the behavior of real video data.

3.5 Performance Metrics

In this stage, we describe the performance metrics of the simulation that were used to evaluate FEC performance. We can validate them by getting the packet loss from the simulation results, and calculating packet loss ratio depending on lost packets and received packets. These performance metrics were used by other researchers [27, 28] where these performance metrics were chosen carefully to represent FEC performance.

3.5.1 Packet loss

By comparing the sent packet amount at the sender–side, and the received packet amount at the receiver-side, we can easily obtain the packet loss amount. The lost packets refer to the packets that failed to reach their destination. As a result of packet loss amount the performance of FEC would vary, because as we showed previously that there is a correlation between them, during our research we computed the lost packet at (r1) for CBR over FEC, and the received packets at the FEC receiver (R2), to get the effect of the FEC parity amount.

3.5.2 Packet loss ratio (PLR)

Depending on the calculated packet loss we can obtain the packet loss ratio by using equation 3.1, where the packet loss ratio depends on the received and lost packets. We can assume that there is a correlation between packet loss and the packet loss ratio, which will have a correlation with FEC performance.

$$PLR = \frac{\text{lost packet}}{\text{lost packet} + \text{received packet}} \tag{3-1}$$

3.6 Simulation Phase

In Chapter 4 we go through stage four of our research methodology, where the simulation model, performance metrics and environment (scenarios, topology and configuration) are put together using network simulator input scripts and then run. The results that were collected by running the simulator under specific circumstances showed the suitable FEC redundant packet size to enhance the video streaming quality in the presence of packet loss. In order to get good results we had to run the simulator several times for each simulation, then took the average for all simulation we did.

To introduce randomness in a simulation model, a random number generator (RNG) was used by Ns-2. RNG is an important core component of the Ns-2. RNG generates random numbers using computational algorithms, and these numbers are determined by a shorter initial value known as a seed; this seed identifies the starting location where RNG starts to pick numbers to produce random results; otherwise the results would be the same for all simulations. For our experiment each simulation was run 30 times using different *Random Number Generator (RNG)* seeds to get the averaged results. In order to validate our simulation, we ran our simulator with the input that was used in [27, 28] until we obtained the same results that they had in the same experiments; with this procedure we can ensure that our simulation ran well.

3.7 Simulation Process

In stage five of our research methodology, we introduce in Chapter four. We collected the output data from the simulation phase in this chapter. As a result of our input there is an output, a *trace file* that consists of twelve columns, and shows complete information about every individual event that happened in the simulation phase.

3.8 Results Analysis and Data Interpretation

In Chapter five we go through phase six of our research methodology. This stage undertakes analysis of the results gained from the simulation. It discusses, evaluates and interprets the results gained.

We use the data collected in phase five as an input to a graphical simulation display tool, in order to analyze the trace file, we can use one of the programming languages that were designed for processing text-based data; for our project we use (awk). [53] Awk has been a part of UNIX since 1978, and the name awk comes from the last names of its creators Alfred *Aho*, Peter *Weinberger*, and Brian *Kernighan*. With awk, trace files are treated as a sequence of lines, and each line is broken up into a sequence of words.

We can see that an AWK program is a series of pattern-action pairs, written as (pattern {action}), where *pattern* is an expression and *action* is a series of commands. In the procedure of processing the trace file with awk, each line of the trace file is tested against all the patterns line by line, so the awk action is executed for each expression that is true, and through that tool we can see our data presented in a graphical mode.

3.9 Results Presentation

Stage seven of our research method, we introduce in Chapter five. This phase is very important, first, because it is the final stage that the project has to move into. Secondly, this stage involves presenting and showing the results from the simulation. Our conclusion, depending upon the analysis of the collected results, and finally the contribution of this project is presented Chapter 6.

3.10 Summary

In this chapter, the method that was suggested for this study was presented. It was carefully selected and employed to fit the requirements, problem statement, and we see that simulation was the ideal tool for our project. Throughout the phases, step by step, we notice that each phase leads to the next one. And for each phase we use the appropriate demand for it; NS-2 was chosen as the network simulator for this research, we started with the definition of the experimental objectives and goals of this simulation, followed by a definition of simulation, and how carefully we chose our performance metrics, and in the simulation phase how to combine the simulation model, performance metrics and environment to run our simulation followed by collecting our output and analyzing it and formulating the experimental conclusions.

Chapter 4

Simulation Phase and Process

4.1 Introduction

In the previous chapter, the research methodology was presented. The methodology consists of seven phases that includes; phase one, the experimental objectives; phase two, the simulation model and environment; phase three, definition of the performance metrics, followed by phase four, the simulation; phase five, the simulation process; phase six, analysis of the results and data interpretation, finally, in the last phase was the presentation of results. The phases or components complement each other in a timely way; each phase comes in the right place and time until the last phase is finished.

Through this chapter we continue our methods; after we introduced the first three phases in Chapter three, we introduce in this chapter phase four and five, which are the simulation phase and the simulation process. In the simulation phase, the simulation was run for each parameter 20 times in order to ensure that the results obtained were accurate, where we assume that the results in the beginning of the simulation experiment were not accurate, actually, to make our result more confident the simulations were run many times using RNG. In the simulation process, we collected the output data, which was a result of our inputs after a limited amount of time. With the Ns2 simulator, the output of the simulation is raw data in the form of a *Trace-File*; this raw data is more useful after analysis, which we apply in the next chapter.

4.2 FEC Performance

As we outlined in section 3.5, FEC performance can be evaluated by the performance metrics of the simulation; we can validate these performance metrics by calculating the packet loss and packet loss ratio. These performance metrics were used by other researchers [25,26], and as we mentioned in section 2.3, the main reason behind choosing FEC is to enhance the quality of real-time video applications in the presence of packet loss. So before we start to describe our experiment environment, it is worthwhile to list the relations between these metrics and parameters, that were used in our simulation experiment, to enhance video streaming in the presence of packet loss.

4.2.1 FEC Performance Metrics

FEC blocks consist of serial number of packets; this number of packets determines the block size. An FEC packet is the unit of data of any object, where the number of bytes in this packet determines the packet size. [27], Within the FEC encoding, if we have *k* packets as the source data, after the FEC encoding it will be *n* packets, where $n>k$ and $n-k$ =the redundant data that was added by the encoder from the original data.

It is important for us to identify what parameter affects the FEC performance, Because not all parameters affect the FEC performance [38], for our project, our work is related to three important variables: FEC block size, FEC packet size and FEC parity amount. In [24], the experiment results showed that with small packet sizes, when they used FEC over UDP in the video transmission, there was a reduction in packet loss. Either in [27,28], it is clear that when the FEC block size is 200 packets, and each packet contains 1100 bytes, this introduced the minimum amount of packet loss. So the three experiments gave the same results, that to have the minimum amount of packet loss, you have to search for the appropriate FEC packet size and FEC block size. These results lead us to investigate the third variable, which enhanced the three experiment results, FEC parity amount.

4.2.2 FEC Parity Amount

For our simulation we use the results in these experiments [27,28], in order to investigate and evaluate which parity amount is the appropriate parity amount that enhanced the results in these experiments, to achieve a high performance level for FEC. In the case we do achieve that, we guarantee a high level of presentation of the transmitted video. So in this chapter, we vary the amount of parity to evaluate the performance of FEC over the Internet.

In order to validate the efficiency of the simulated FEC parity amount, we make comparisons between the packet amount that was sent from the sender–side, and the received packet amount at the receiver-side; we can easily obtain the packet loss amount at the first router. The loss packet refers to the packets that fail to reach their destination target. As a result of packet loss amount the performance of FEC will vary, because as we showed previously there is a correlation between them. During our research we computed the lost packet at (r1) for CBR over FEC as it shows in figure 4.1, and received packets at FEC receiver (R3), to get the effect of FEC parity amount.

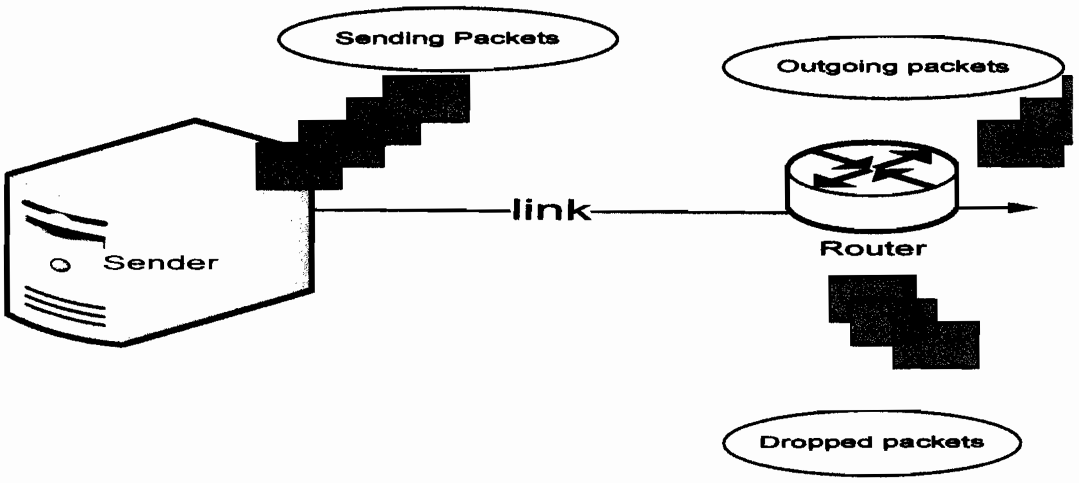

Figure 4.1: Lost Packets at the first Router

Depending on the packet loss we can compute the packet loss ratio by using Equation 3.1 in section 3.1, where the packet loss ratio depends on the received and lost packet, we can assume that there is a correlation between packet loss and the packet loss ratio, which finally will have a correlation with FEC performance. The reason behind computing the packet loss ratio is to enhance the results we obtained by using packet loss at the first router, where with packet loss calculation, we just use the lost packets; in the packet loss ratio we calculate packets that were received at the receiver-side and compare them with the packets lost at the first router, and the sent packets from the sender-side as it shows in Figure 4.2.

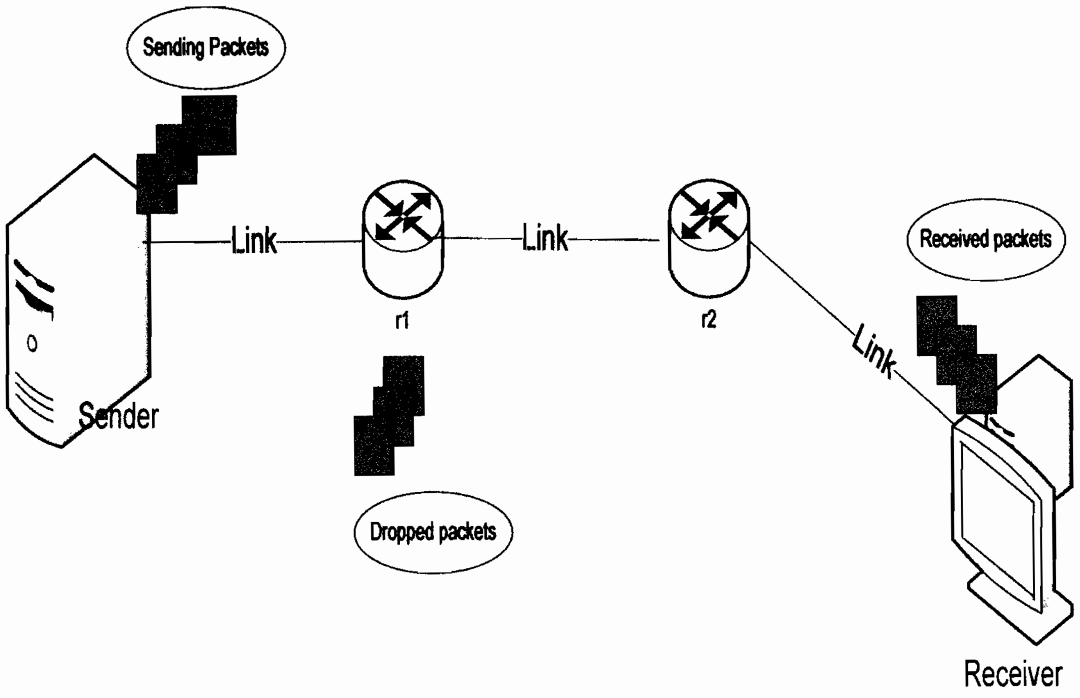

Figure 4.2: Sent and Received Packets

4.3 Experiment

4.3.1 Experimental Environment

Our experiment was done with a simulation environment using network simulator 2. Throughout this environment we chose the experimental events, parameters and objects close to real life, to give our experiment more reality. We setup and ran the required topology using the programming language OTcL script, and initiated an event scheduler that tells the traffic sources when to start transmitting packets, where these packets were not real packets; which means that these packets were not from a real network.

Through applying our proposed topology we used the available protocols in the Ns-2 such as IP, TCP and UDP, and we used traffic source behavior like FTP and CBR, and we used Drop Tail as the router queue management policy. After the end of each simulation process, Ns2 produced a trace-file as an output, which we used to analyze the data to obtain our results.

4.3.2 Experiment Setup

In section 3.4 we mentioned our scenario in general; in this section we make more details about our topology before we start our experiment. We used here Reed Solomon code to model the FEC, where this code adds redundant data to the original data, this data used by the receiver to reconstruct the received video. Through running the simulation we change the parity amount, so we start with 5 packets then 10 packets up to 35 packets, where we added 5 more packets in each different simulation, the idea behind this variation in parity amount is to find the suitable one that guarantee minimum amount of packet loss.

The simulation scenario is depicted in figure 4.3. We designed the topology with eight nodes; the reason behind this number of nodes was to make our topology simpler and more efficient, and during this scenario we can easily collect our data for the analysis phase. We are interested to collect the data on the behavior of Sender3, the first router (r1) where the drop of data occurs, and Receiver 3. Our scenario consists of six end-systems (S1,S2,S3,R1,R2 and R3) communicating through a network of two routers r1 and r2; each router implemented has a limited queue set to 25 bytes, and the two routers were configured using *Drop Tail* management.

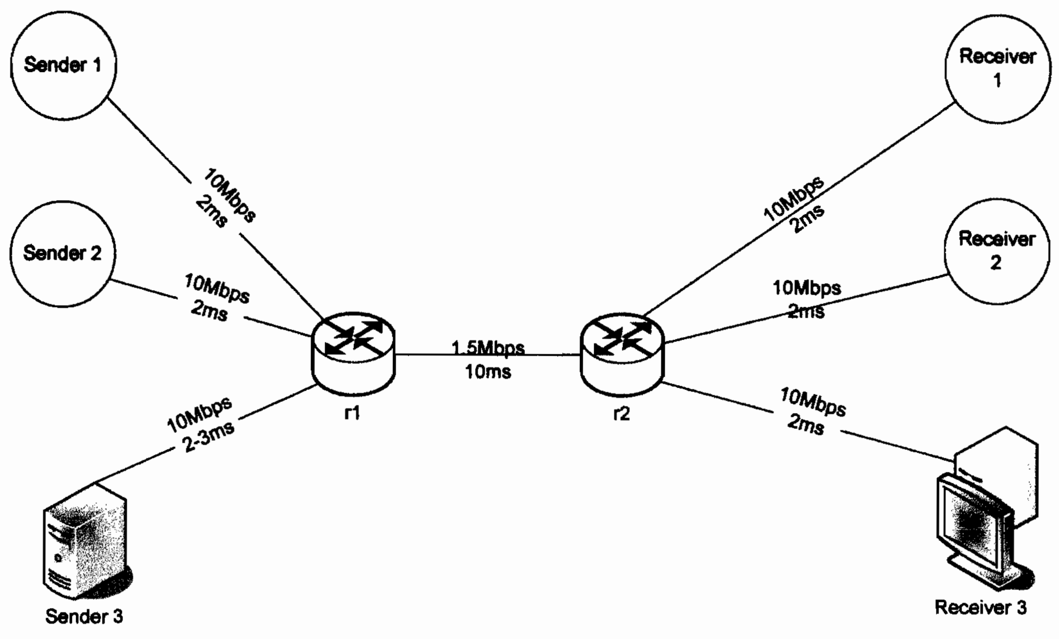

Figure 4.3: Simulation Topology

The links between end system 1, end system 2 and end system 3 and the first router r1 is 10 Mbps. And the same speed between the second router r2 and end system 4, end system 5 and end system 6. The link between the two routers was set to 1.5Mbps. The link delay in the link between the end systems S1, S2, R4, R5, and R6, and the two routers was set to 2 ms. The link delay between end system S3 and the first router was set to 2-3 ms, where the delay during the simulation processes was chosen randomly by using a random number generator (RNG) seeded to the Ns2, in order to give more reliability for the results, and the link delay between the two routers was 10 ms.

The simulation scenario was set to transmit both real-time video and elastic applications over IP, and because of that, in this experiment we used both transport protocols UDP and TCP, the reason behind using TCP was to send FTP over the network, and to make congestion at the first router in order to make data losses occur.

During implementation of this scenario we made S1 and S2 send FTP traffic where the data flowed through the network to R1 and R2, and S3 send CBR traffic where the data flowed through the network to R3. We chose CBR as real-time data because it generated constant traffic during the simulation and because of the benefit of CBR for the real-time video content in limited channel capacity like our topology, Figure 4.4 depicts the network topology.

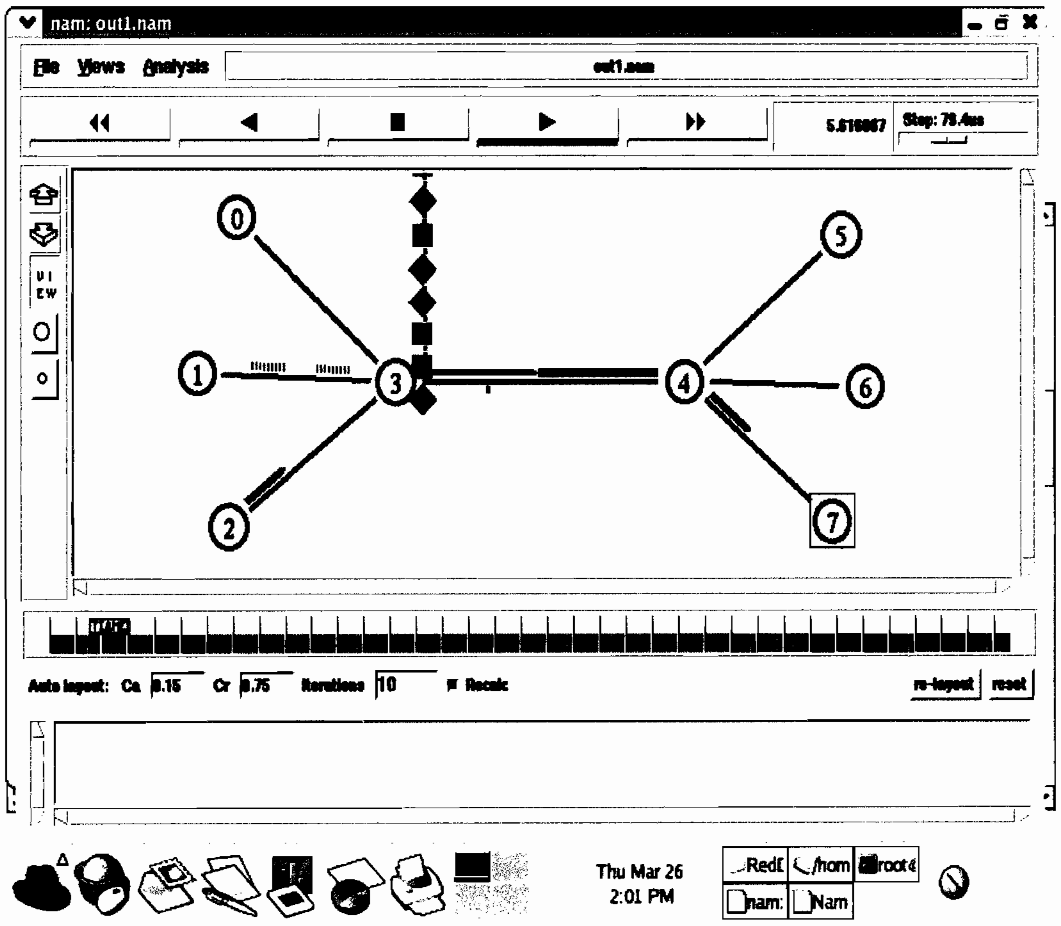

Figure 4.4: Network Topology

The number 0 refers to S1, 1 refers to S2, 2 refers to S3, 3 refers to r1, 4 refers to r2, 5 refers to R1, 6 refers to R2, 7 refers to R3. The blue color refers to FEC and the black color refers to FTP. Sender S1, which was represented as 0 in the simulation topology sent FTP, and Sender S2, which was represented as 1 in the simulation topology sent FTP, and the sender S3 which represent as 2 in the simulation topology sent CBR, and the receivers R1 and R2 received FTP data, where R3 received FEC data.

4.3.3 Experiment Parameters

As we outlined at the beginning of this project we adopted the experiment handled by the researcher in [27,28], so for our proposed scenario we adopt the same parameters, the major idea behind that is to make no change in the results that were collected by the researchers, and to extend their results by looking for the appropriate FEC parity amount that would enhance their work, where in the end we can increase the FEC performance level, through decreasing the amount of lost packets for the transmitted video, which finally guarantees a high quality of video presentation at the receiver-side. Table 4.1 shows the inserted parameters that we used in our experiment.

Table 4.1: Simulation Parameters

Parameters	Value
FEC packet size	1100 bytes
FEC block size	200 packets
FEC parity amount	5,10,15,20,25,30,35
FEC block number	50
FTP packet size	1000 bytes
Side link bandwidth	10 Mbps
Side line delay for S1,S2,R1,R2,R3	2ms
Side line delay for S3	2-3ms
Bottleneck bandwidth	1.5 Mbps
Bottleneck delay	25ms
Management Policy	Drop Tail
Queue limit	25 bytes
Simulation time	100 seconds

4.3.4 Experiment Execution

After we inserted all events and parameters; we executed the simulation for each parity amount 20 times; we started with a parity amount of 5 packets, then for the next simulation we used 10 packets as redundancy, and we continued to add 5 packets for each simulation after that up to 35 packets as parity amount. We ran each simulation for a sufficient duration of time as we showed in the simulation parameters, in order to get good results and to allow for loss to occur during the transmission process. We did not take any results for the first five seconds, to guarantee the accuracy of the gathered data. After each simulation process, and as a result of the inserted events and parameters, the output was collected as a trace-file; these results are shown in the next section.

4.4 Simulation Results

After we ran the simulation up to the time that we constrained the simulation, the results were the most important thing that we wanted to obtain, unfortunately Ns2 does not provide any tools for getting results. Instead, Ns2 provides a trace file that consists of 12 fields representing all the events taking place during the time period of each simulation run. The trace file size of Ns2 is not fixed and it varies with respect to many factors; the number of nodes, the number of transport agents, and the simulation time, hence, if the size of the trace file is too big, it will make the analysis of it more complex. For this reason we made our topology easy and with no additional nodes, whereby this way we guarantee simplicity and efficiency of our output. But how to solve the problem of getting results, because without results we cannot get tables and cannot analysis the data.

In order to analysis the trace file, we can use one of the programming languages that designed for processing text-based data; for our project we use (awk). Awk has been a part of UNIX since 1978. With awk trace files are treated as a sequence of lines, and each line is broken up into a sequence of words. In the procedure of processing the trace file with awk, each line of the trace file is tested against all the patterns line by line, so the awk action is executed for each expression that is true. Table 4.2 shows the fields in the raw trace file, after we analyzed the trace-file using awk; now we can put our output data into tables where we can do the data analysis to obtain results and conclusions.

Table 4.2: Fields in the Raw Trace-File

	Fields of Trace-File	Data meaning
1.	Event	The event that happens during the simulation.
2.	Time	The time of the event.
3.	From Node	The origin source of the packet denoted by the ID.
4.	To Node	The destination of the packet denoted by the ID
5.	Packet Type	The type of packet.
6.	Packet Size	The packet size (in bytes).
7.	Flags	Ns2 implements only the Explicit Congestion Notification (ECN) bit
8.	Flow Id	Flow identity is specified at IP packet header.
9.	Source Add	The source address is specified at IP packet header.
10.	Dest. Add	The destination address is specified at IP packet header.
11.	Seq. Num	The sequence number of the packet and it is specified by a transport layer protocol.
12.	PacketId	The packet identity.

We can see that the trace file starts with an event descriptor; we summarize these descriptors in Table 4.3, where the event appears in one of the symbols as is illustrated in Table 4.3.

Table 4.3: Symbols in the Raw Trace File

Symbol	Symbol Name	Description
+	Enque	Represents packet arrival at queue.
-	Deque	Represents packet departure from the
d	Drop	Represents packet drop.
r	Receive	Represents the packet reception at the destination.

In the second field [53] the simulation time in the raw file field is represented as *ms,* and in the third and fourth field the source and destination node are represented by a number, in the fifth field the packet type is represented by *cbr* or *ack* etc., followed by the size of the packets in bytes in field six, flag field is in the seventh field as represented with seven digits "------". Ns2 implements only the Explicit Congestion Notification (ECN) bit, and the remaining bits are not used. The next field is flow id (fid) of IP that can be set for each flow. The source and destination address are set in the next fields. The next field is the network layer protocol packet sequence number, with the unique id of the packet set in the last field.

. Now, and after using awk we can easily collect our tables for each simulation, as we see in the following tables, where it shows the results of the simulation using 5,10 ,15, 20, 25, 30 and 35 packets as redundant data with the original data, where we send 30 blocks as original data, which is equal to 6000 packets. The following tables represent the collected data from the simulation runs; we ran the simulation 20 times for each proposed parity amount.

Table 4.4: Results of Simulation Using 5 Packets as Redundancy

Original packets: 6000	
Parity amount: (5*30)	
Lost packets	Received packets
823	5327
826	5324
831	5319
799	5351
816	5334
819	5331
816	5334
823	5327
806	5344
840	5310
817	5333
833	5317
824	5326
820	5330
839	5311
828	5322
828	5322
818	5332
810	5340
841	5309

In this table we used 5 packets for each block as the parity amount, and other variables were fixed as was shown in Table 4.1.

Table 4.5: Results of Simulation Using 10 Packets as Redundancy

Original packets: 6000	
Parity amount: (10*30)	
Lost packets	Received packets
834	5466
836	5464
855	5445
830	5470
837	5463
861	5439
841	5459
835	5465
833	5467
835	5465
832	5468
844	5456
831	5469
847	5453
848	5452
835	5465
839	5461
839	5461
859	5441
827	5473

In this table we used 10 packets for each block as the parity amount, and other variables were fixed as was shown in Table 4.1.

Table 4.6: Results of Simulation Using 15 Packets as Redundancy

Original packets: 6000	
Parity amount: (15*30)	
Lost packets	Received packets
852	5598
857	5593
863	5587
856	5594
865	5585
862	5588
881	5569
851	5599
853	5597
878	5572
862	5588
858	5592
858	5592
858	5592
844	5606
854	5596
855	5595
865	5585
862	5588
855	5595

In this table we used 15 packets for each block as the parity amount, and other variables were fixed as was shown in Table 4.1.

Table 4.7: Results of Simulation Using 20 Packets as Redundancy

Original packets: 6000	
Parity amount: (20*30)	
Lost packets	Received packets
867	5733
878	5722
884	5716
869	5731
886	5714
881	5719
881	5719
901	5699
899	5701
859	5741
881	5719
897	5703
879	5721
865	5735
888	5712
883	5717
880	5720
881	5719
875	5725
875	5725

In this table we used 20 packets for each block as the parity amount, and other variables were fixed as was shown in Table 4.1.

Table 4.8: Results of Simulation Using 25 Packets as Redundancy

Original packets: 6000	
Parity amount: (25*30)	
Lost packets	Received packets
917	5833
898	5852
903	5847
893	5857
898	5852
894	5856
911	5839
897	5853
891	5859
905	5845
899	5851
893	5857
909	5841
898	5852
889	5861
920	5830
898	5852
888	5862
903	5847
897	5853

In this table we used 25 packets for each block as the parity amount, and other variables were fixed as was shown in Table 4.1.

Table 4.9: Results of Simulation Using 30 Packets as Redundancy

Original packets: 6000	
Parity amount: (30*30)	
Lost packets	Received packets
936	5964
919	5981
902	5998
904	5996
913	5987
911	5989
912	5988
922	5978
947	5953
913	5987
910	5990
922	5978
915	5985
914	5986
918	5982
976	5924
919	5981
936	5964
938	5962
912	5988

In this table we used 30 packets for each block as the parity amount, and other variables were fixed as was shown in Table 4.1.

Table 4.10: Results of Simulation Using 35 Packets as Redundancy

Original packets: 6000	
Parity amount: (35*30)	
Lost packets	Received packets
930	6120
946	6104
952	6098
939	6111
934	6116
927	6123
946	6104
944	6106
930	6120
951	6099
930	6120
943	6107
939	6111
928	6122
938	6112
932	6118
927	6123
943	6107
931	6119
925	6125

In this table we used 35 packets for each block as the parity amount, and other variables were fixed as was shown in Table 4.1.

4.5 Conclusion

In this chapter, we presented the results of the simulation experiment for FEC in sending video as a real-time application, where we used fixed sizes for the FEC block size and FEC packet size; in addition we varied the amount of parity to investigate the appropriate parity amount that would guarantee a high level of FEC performance through decreasing the amount of packet loss. The appropriate parity amount caused the least amount of packet loss, by decreasing the amount of lost packets through transmission; the amount of received packets would increase, so by increasing the amount of received data the quality of video presentation would increase, which means that the performance level of FEC would increase. In the next chapter we will analyze the results to evaluate the appropriate parity amount from the results we collected in this chapter.

Chapter 5

Result Analysis and Presentation

5.1 Introduction

In this chapter we discuss the performance evaluation process of the FEC, through the use of FEC as an error correction mechanism in transmitting real-time applications over the Internet. This evaluation was done with the simulation technique; the performance evaluation aims to identify how the parity amount affects the packet loss during video transmission over the Internet. The appropriate parity amount that was seeded with the original data that guarantee the least lost packets of transmitted video is the thing we were looking for as an output of this research. Section 5.2 undertakes the analysis of the results gathered in the last chapter. Section 5.3 presents the results and gives an insight into the system. Section 5.4 summarizes the chapter.

5.2 Results Analysis and Data Interpretation

5.2.1 Packet Loss

In Chapter 4 we presented the results as we collected it from the simulation, in Table 5.1 we calculated the mean for the lost packets and received packets for each simulation; we also computed the mean for each table mentioned in the last chapter, with respect to the parity amount that was used to obtain the results from the simulation. We used equation 5.1 to compute the mean of lost packets and received packets, which was the average of the lost and received packets, and we can easily compute this by summing all the values for each simulation output (either lost packets or received packets), then divided the sum by the number of runs for the simulation (20 runs).

By using the equation (5.1) below, we obtained the results shown in Table 5.1, where we used 5 packets as parity amount; the remainder of the results for the other parity amounts are listed in Appendix A.

Table 5.1: Lost Packets and Received Packets (5 Packets as Parity Amount)

Lost packets	Received packets
823	5327
826	5324
831	5319
799	5351
816	5334
819	5331
816	5334
823	5327
806	5344
840	5310
817	5333
833	5317
824	5326
820	5330
839	5311
828	5322
828	5322
818	5332
810	5340
841	5309
822.85	5327.15

$$\bar{X} = \frac{\sum_{i=1}^{i=n} xi}{n} \qquad (5.1)$$

Where:

$\bar{X}$: is the average.

xi: is the result of each run (lost packet, or received packet).

n : is the number of runs.

After having calculated the average of lost packets and received packets for each simulation for the 20 runs we gathered the results using Table 5.2 below

Table 5.2: The Average of Lost Packet and Received Packet

Original sending packets	Parity sending amount	Average of Lost packets	Average of Received packets
6000	150	822.85	5327.15
6000	300	839.9	5460.1
6000	450	859.45	5590.55
6000	600	880.45	5719.55
6000	750	900.05	5849.95
6000	900	921.95	5978.05
6000	1050	936.75	6113.25

We can observe from this table that the number of original sending packets is fixed (6000), and we varied the amount of parity sent in each simulation. We can observe that as the amount of parity increased, the amount of lost packets increased, the reason behind the increase in lost packets is that as the transmitted data increases the traffic would also increase, which causes more congestion, and as a result of this congestion the lost packets would become greater and the received packets would decrease. The increase in packet loss would affect the performance of FEC and vice versa, so the level of FEC performance would increase as the lost packets decrease.

5.2.2 Results Validity

For this research, it is important to be sure of each step that we accomplish. In this research we simulated a limited number of runs (20 runs), with respect to a limited interval of parity (5-35 packets/Block) during the journey of looking for the appropriate parity amount, and in order to give the results more *credibility*, we will ensure the mean for all gathered results by computing the confidence interval. The reason behind using the confidence interval, because it provides the mean results with a certain probability of containing the true mean, where the confidence interval is a range around the computing mean.

After computed the means of lost packets and received packets for each simulation runs, we can calculate the variance values related to each mean [59], it provides a description for the collected distributed values (lost packets values and received packets values). Equation 5.2 shows how to compute the variance values. Depending on the variance values, we calculated the standard deviation values; the standard deviation gives the variability for the collected lost packets (or received packets) set. We calculated standard deviation values for each of the variance values by using Equation 5.3. We noticed that there is a relation between variance values and standard deviation, where the standard deviation equals the square root of the variance values; then we calculated the standard error depending on the standard deviation values by using Equation 5.4. The standard error values actually depend on the number of runs for each simulation, where the standard error became smaller as we increased the number of runs, and depending on standard error we finally derived the margin of error, or confidence interval, by using Equation 5.5; the collected data is shown in Table 5.3 and Table 5.4 as an extension of Table 5.2 and using the above equations.

$$S_M = Avg[(xi - \bar{X})^2] = \frac{[(xi-\bar{X})^2]}{n-1} \qquad (5\text{-}2)$$

$$S = \sqrt{Avg\,[(xi - \bar{X})^2]} = \sqrt{\frac{\sum_{i=1}^{i=n}[(xi-\bar{X})^2]}{n-1}} \qquad (5\text{-}3)$$

$$\sigma M = \frac{S}{\sqrt{n-1}} \qquad (5\text{-}4)$$

$$\text{Margin of Error} = S * \text{t\_value} \qquad (5\text{-}5)$$

Where :

$\bar{X}$: is the average.

xi : is the result of each run (lost packet, or received packet).

n : is the number of runs.

S_M: is the variance.

Avg: is the mean.

S: is standard deviation.

σM: is the standard error.

We found the confidence interval, or margin of error, which gives us a range where the true mean is within a certain probability; this value must be determined before calculated the confidence interval. In this research we used the probability value of 95%, which means that for any number of runs of any simulation, the mean of the results can be found around the mean value with a 95% degree of probability; the t_value being determined by default [59], when we use 95% as a probability then the t_value is set to 2.01.

Table 5.3: The Confidence Interval of Lost Packets

Sent Parity Amount	Average of Lost Packets	Variance Values	Standard Deviation Values	Standard Error Values	Confidence Interval Values
150	822.85	121.6078947	11.02759696	2.465845643	5.161074235
300	839.9	92.51578947	9.618512851	3.101372737	6.491247725
450	859.45	73.83947368	8.592989799	1.921450932	4.021643011
600	880.45	117.2078947	10.82625950	2.420825218	5.066845402
750	900.05	76.26052632	8.732727313	1.95269719	4.087042181
900	921.95	299.3131579	17.30066929	3.868547259	8.096962451
1050	936.75	71.25000000	8.440971508	1.887458609	3.950496261

After using the above equations, we finally derived the confidence interval as it shows in Table 5.4; we extended it from Table 5.3, which shows the mean ± the margin of error for lost packets.

Table 5.4: The Average of Lost Packets and Margin of Error

Average of Lost Packets	Confidence Interval Values
822	±5
839	±4
859	±4
880	±5
900	±4
921	±8
936	±3

Table 5.5: The Confidence Interval of Received Packet

Sent Parity Amount	Average of Received Packets	Variance Values	Standard Deviation Values	Standard Error Values	Confidence Interval Values
150	5327.15	121.6078947	11.02759696	2.465845643	5.161074235
300	5460.1	92.51578947	9.618512851	2.150764858	4.501602573
450	5590.55	73.83947369	8.592989799	1.921450932	4.021643011
600	5719.55	117.2078947	10.82625950	2.420825218	5.066845402
750	5849.95	76.26052632	8.732727313	1.95269719	4.087042181
900	5978.05	299.3131579	17.30066929	3.868547259	8.096962451
1050	6113.25	71.2500000	8.440971508	1.887458609	3.950496261

In the same manner, by using the above equations, we finally get the confidence interval as it shows in Table 5.6; we extend it from Table 5.5, as it shows the mean ± the margin of error for received packets.

Table 5.6: The Average of Received Packets and Margin of Error

Received Packets	Confidence Interval Values
5327	±5
5460	±4
5590	±4
5719	±5
5849	±4
5978	±8
6113	±3

5.2.3 Packet Loss Ratio (PLR)

As we outlined in Chapter 4, we ensure the FEC performance by using two main factors, which are packet loss (PL), and packet loss ratio (PLR), in section 5.2.1 we investigate PL for each run on the proposed topology, in this section we compute packet loss ratio, which gives the loss ratio with respect to lost packets and received packets.

We can compute the packet loss ratio by using Equation 5.6. From Table 5.2 we obtained the parity amount, lost packets and received packets from the results gathered in Chapter 4.

To make our calculation serve the goal of this research, we compute PLR for two cases, the first one we calculate PLR before using FEC, and PLR after using FEC, the idea here is to differentiate between the amounts of lost packet in both cases. This will enhance the results that we hope we will get to enhance the efficiency of FEC.

$$PLR = \frac{\text{lost packet}}{\text{lost packet} + \text{received packet}} \qquad (5\text{-}6)$$

Where: PLR is the packet loss ratio.

According to the results gathered in Chapter 4, we calculated the PLR after using FEC and before using FEC for each run within each simulation, and then we calculated the PLR average for the 20 runs for the same parity amount by using Equation 5.1, as it shows in Table 5.7 (as an example, using a parity amount of 5 packets). The rest of the results are shown in Appendix B.

Table 5.7: PLR Before Using FEC and PLR After Using FEC (with the parity amount = 5 packets)

Loss packet	Received packet	PLR Before	PLR After
823	5327	0.137166667	0.133821138
826	5324	0.137666667	0.134308943
831	5319	0.1385	0.135121951
799	5351	0.133166667	0.129918699
816	5334	0.136	0.132682927
819	5331	0.1365	0.133170732
816	5334	0.136	0.132682927
823	5327	0.137166667	0.133821138
806	5344	0.134333333	0.131056911
840	5310	0.14	0.136585366
817	5333	0.136166667	0.132845528
833	5317	0.138833333	0.135447154
824	5326	0.137333333	0.13398374
820	5330	0.136666667	0.133333333
839	5311	0.139833333	0.136422764
828	5322	0.138	0.134634146
828	5322	0.138	0.134634146
818	5332	0.136333333	0.13300813
810	5340	0.135	0.131707317
841	5309	0.140166667	0.136747967
822.85	**5327.15**	**0.137141667**	**0.133796748**

After we calculated PLR before and after using FEC through the results in Table 5.2, we can generate a new Table 5.8, which holds the new values, and represents the mean of the packet loss ratio after using FEC, and the mean of the packet loss ratio before using FEC for each simulation with respect to the parity amount.

Table 5.8: The Mean of PLR Before and After Using FEC.

Parity amount	Lost packets	Received packets	PLR before FEC	PLR after FEC
150	822.85	5327.15	0.137141667	0.133796748
300	839.9	5460.1	0.139983333	0.13331746
450	859.45	5590.55	0.143241667	0.133248062
600	880.45	5719.55	0.14674167	0.133401515
750	900.05	5849.95	0.150008333	0.133340741
900	921.95	5978.05	0.153658333	0.133615942
1050	936.75	6113.25	0.156125000	0.13387234

By sending redundancy data with original data using FEC, we can observe that as the parity amount increases the lost packets increase and the received packets decrease; by calculating the PLR after using FEC and as the parity amount increases, we notice that the packets loss ratio increases as well.

The reason behind the PLR increasing is that as we increase the redundancy amount transmitted over network it requires more bandwidth, furthermore because we are using fixed bandwidth this will cause more traffic on the link, and would cause congestion to occur; when congestion occurs, the packet loss will increase, which means that received packets will decrease, and as a result the PLR will be smaller.

Furthermore if we make a comparison between PLR in the two cases, before and after using FEC, we notice that by using FEC, PLR was less than the PLR when we did not use FEC; this result can contribute to increasing the performance of FEC, because when the PLR decreases it means that the packet loss decreases, which causes an increase in the video presentation quality, this points to the efficiency of using FEC in transmitting real-time applications over the Internet.

After we calculated the mean of the PLR before and after using FEC, and to give our results more credibility we calculated the confidence interval by using the same equation mentioned above in section 5.2.1; Tables 5.9 and 5.10 show the results, respectively, that we gathered from using these equations, for PLR before and after using FEC.

Table 5.9: The Confidence Interval of PLR before Using FEC

Lost packets	Received packets	PLR before FEC	Confidence Interval
822.85	5327.15	0.137141667	±0.000860179
839.9	5460.1	0.139983333	±0.000750267
859.45	5590.55	0.143241667	±0.000670274
880.45	5719.55	0.14674167	±0.00084447
900.05	5849.95	0.150008333	±0.000681174
921.95	5978.05	0.153658333	±0.001349494
936.75	6113.25	0.156125000	±0.000658416

Table 5.10: The Confidence Interval of PLR after Using FEC

Lost packets	Received packets	PLR after FEC	Confidence Interval
822.85	5327.15	0.133796748	±0.000839199
839.9	5460.1	0.13331746	±0.00071454
859.45	5590.55	0.133248062	±0.000623511
880.45	5719.55	0.133401515	±0.000767704
900.05	5849.95	0.133340741	±0.000605488
921.95	5978.05	0.133615942	±0.001173473
936.75	6113.25	0.13387234	±0.00056035

5.3 Presentation of results

We conducted this simulation experiment to investigate and evaluate the effect of different FEC parity amounts against the amount of packet loss and the packet loss ratio, where the final goal is to propose an appropriate parity amount that would guarantee a high level performance for FEC, when it is used as an error correction mechanism in transmitting real-time applications like video streaming over the Internet.

We presented in the last section the results that we gathered from running the simulation during specific conditions, and in this section we present the complete results in tables and graphics related to the same summarized results, where we can present our proposed appropriate FEC parity amount for this research.

Table 5.10 presents the result of the simulation experiment; it is clear that we used different FEC parity amount (5 packets up to 35 packets), we can observe from the values of packet loss that as the parity amount increases the number of lost packets increase, either because when we increase the parity amount it means that we increase the traffic over the link; this will cause congestion to occur, and as a final result the number of lost packets will increase as well. On the other hand, as the lost packets increase, the amount of received packets will decrease, so the constructed video at the destination will not be complete, and in this situation the video presentation quality will not match the user's needs.

Table 5.11: The Result of the Simulation Experiment

(Redundant data Vs packet loss)

Redundant data	PL
5	822
10	839
15	859
20	880
25	900
30	921
35	936

To give a clearer explanation of these results, we used the cropped results to plot a graph using the data in Figure 5.1.

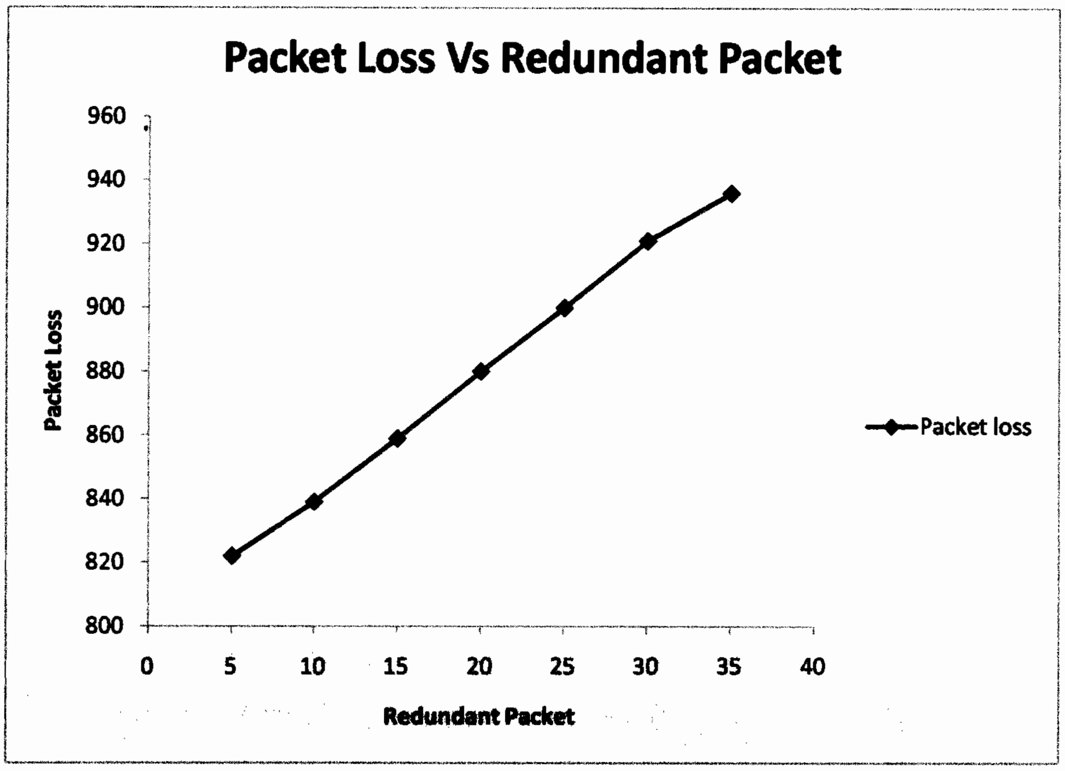

Figure 5.1: Redundant Packet vs. Packet loss

It is clear from Figure 5.1 that from the results that we gathered in Table 5.10, we can see that the least amount of packet loss is 822, which occurs when the value of the redundant data is the least, which is 5 packets for each sending FEC block, and the packet loss increased as the parity amount increased, because of the increase in overhead, which causes more traffic, and as a result congestion will be more, and the packet loss would increase.

Furthermore, to give our collected results more credibility, we used Equation 5.6 to calculate the packet loss ratio, which enhanced our results in Table 5.10, and in Figure 5.1, because by using this equation we joined two variables together to make a better comparison of our results, so we used lost packets at the first router, and received packets at the receiver-side to ensure our final proposed results that shows the most appropriate parity amount; we calculated the PLR after using FEC and before using FEC, Table 5.11 presents the results.

Table 5.12: The Result of the Simulation Experiment

(Redundant data vs. Packet Loss Ratio before and after using FEC)

Redundant data	PLR before FEC	PLR After FEC
5	0.137141667	0.133796748
10	0.139983333	0.13331746
15	0.143241667	0.133248062
20	0.146741667	0.133401515
25	0.150008333	0.133340741
30	0.153658333	0.133615942
35	0.156125	0.13387234

We can observe from Table 5.11 that the PLR before using FEC was more than the PLR after using FEC, because the number of received packets when we used FEC is more than the number of received packets when we did not use FEC. Furthermore, we can observe that both values of PLR before and after using FEC increased as the parity amount increased, the reason behind that being as the parity amount increased the overhead increased; as a result of the increasing traffic the lost packets would increase.

Furthermore, to enhance our collected results, and to make it clearer, we use Figure 5.2, to give a clearer explanation of these results; we used the cropped results to plot a graph using the redundant data vs. packet loss ratio.

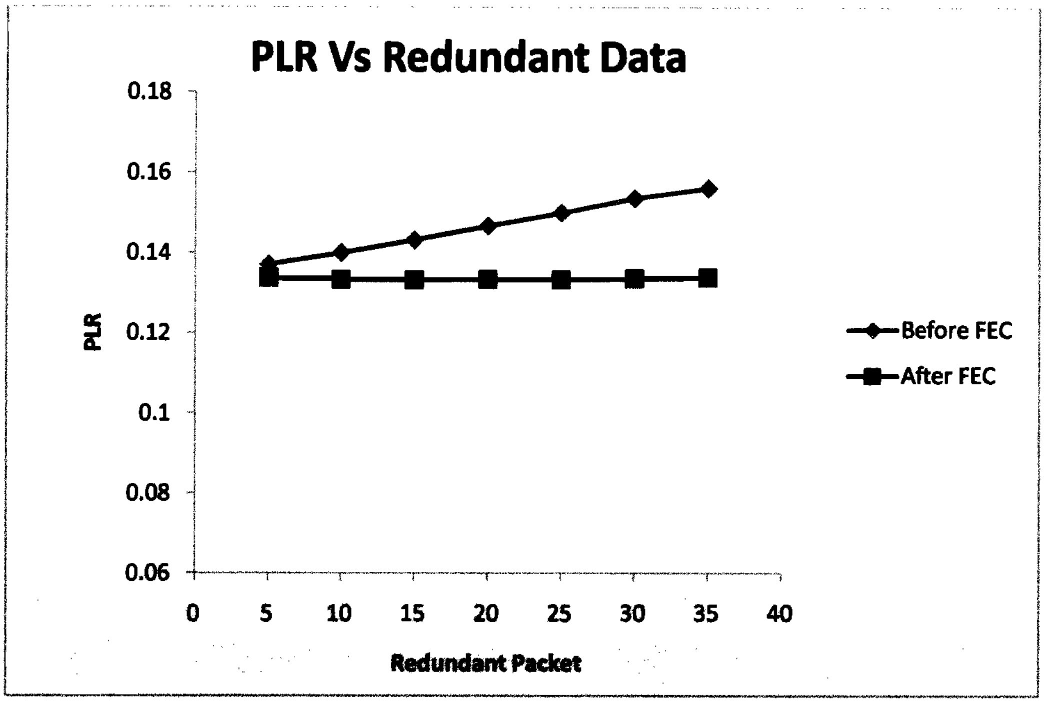

Figure 5.2: Redundant Packet vs. Packet Loss Ratio

We can observe from Figure 5.2 that as the redundant amount increased the benefit of using FEC with the transmitted packets occurred, and the PLR became clearer between using FEC and without using FEC; this proposed result can enhance the idea behind using FEC in the case of transmitting real-time applications over the Internet.

5.4 Experiment Conclusion

In this experiment we investigated and evaluated the performance of using FEC in sending real-time applications like video streaming. We proposed our experiment scenario in a simple and efficient way, using the Ns2 simulator, and by using different parity amounts (5, 10…up to 35 packets), sent with the original data to be used to reconstruct received videos at the receiver-side. Throughout the collected results and after analysis, we gathered our successfully received packets at the destination, and the lost packets at the first router, our proposed solution to enhance the results gathered in [27,28], these results ensure the benefit of using FEC in sending video over the Internet, and shows that FEC is more effective when the parity amount is small, where we found in this experiment that the parity amount that guaranteed the high level performance for FEC as an error correction mechanism was 5 packets for each sending block depending on the results proposed with [27,28].

Furthermore, we showed through this experiment that as the parity amount increased, the packet loss increased, the reason being that as the parity amount increases the overhead increases, this caused more traffic and as a result congestion increased, which led to more packet loss, this in the end would affect the performance of FEC, which finally affects the quality of the video presentation.

5.5 Summary

In this chapter we presented the analysis of the results and evaluated the appropriate parity amount that we obtained from the results that we collected in Chapter 4. We made a comparison between the lost packets and the sending parity amount, through several simulation results; furthermore, we calculated the mean of the lost packets and the confidence interval for each one. After that, and depending on the above results, we calculated PLR before using FEC and after using FEC, and made a comparison between the two results, and finally we presented our proposed solution, by finding the appropriate parity amount that guaranteed the required quality level of FEC, that guaranteed a high level of video presentation for real-time applications. The next chapter will present the conclusions of our research.

Chapter 6

Conclusions and Recommended Further Study

6.1 Introduction

Forward error correction (FEC) is a very important mechanism that enhances the quality of transmitting of real-time application over the Internet in the presence of packet loss. This research was conducted to study the effect of FEC parity amounts, and to investigate and evaluate the most appropriate parity amount sent with the original data using FEC that would guarantee the high level performance of FEC as an error control mechanism.

6.2 Findings

As described in Chapter one, the main goal of this study was to find the appropriate FEC redundant amount that would optimally support video streaming over the Internet.

In the same chapter, we showed that the goal would be to identify the optimal FEC redundant amount to be sent from the server to the client that would have a lower level of packet loss, and would achieve a high range of reconstructed video quality.

We went ahead to list the objectives of this research that were crucial in achieving video streaming with high quality in the presence of packet loss.

Research Objective 1:

To identify the optimal FEC redundant amount that would be sent from source to destination to accomplish perfect compensation in case of packet loss. We managed to identify the suitable FEC redundant amount that will be transmitted with the original data, by using Ns2; this parity amount guaranteed perfect data reconstruction in the presence of packet loss. Results were discussed in Chapter 4 and Chapter 5.

Research Objective 2:

To evaluate the FEC redundant amounts found in the first objective, in order to verify the optimal FEC redundant amount. Through the proposed solution we evaluated the most appropriate parity amount that would guarantee high level performance for FEC, which would finally improve the video presentation quality. Results were discussed in Chapter 4 and Chapter 5.

6.3 Problems and Limitations

6.3.1 Problems

During the phases of this research, we used the Ns2 simulator to aid us in making progress towards achieving our research goal. And because Ns2 provided us with a simple and efficient research environment, we did not encounter any kind of problems in the research looking for the appropriate FEC parity amount.

6.3.2 Limitations

For this research, we implemented a fixed bottleneck bandwidth, which in fact limited the area for the proposed solution. So by increasing this parameter value one could make more progress in the area of enhancing the performance of FEC.

6.4 Contributions

We obtained the following contributions for our research work:

a. Enhanced the results collected in [27, 28] where it was considered as a base for this research.

b. An appropriate FEC redundant amount was proposed in this research, where it could be used to enhance the performance of FEC in the presence of packet loss, for sending videos over the Internet.

6.5 Future Works

It's clear that there will be much work to do for this study to improve the quality of the service; future work could be done in investigating and evaluating the change in bandwidth that could guarantee a better performance for FEC.

References

[1] J. F. Kurose and K. W. Ross, "Computer Networking," A Top-Down pproach, fourth edition, 2007, pp. 615-702.

[2] R. Steinmetz and K. Nahrstedt, "Multimedia Application and Security," Springer Verlag, 2004, pp. 3-15.

[3] S. Tao, J. Apostolopoulos, and R. Guerin, "Real-time monitoring of video quality in IP networks," IEEE/ACM Transactions on Networking, vol. 16, pp. 1052-1065, 2008.

[4] C. Perkins, O. Hodson, and V. Hardman, "A survey of packet loss recovery techniques for streaming audio," IEEE network, vol. 12, pp. 40-48, 1998.

[5] F. Fitzek, A. Kopsel, A. Wolisz, M. Krishnam, and M. Reisslein, "Providing application-level QoS in 3G/4G wireless systems: a comprehensive framework based on multirate CDMA," IEEE Wireless Communications, vol. 9, pp. 42-47, 2002.

[6] R. Puri, K. Lee, K. Ramchandran, and V. Bharghavan, "Application of FEC-based Multiple Description Coding for Internet Video Streaming and Multicast," presented at Proc. Packet Video Workshop, 2000.

[7] K. Kawahara, K. Kumazoe, T. Takine, and Y. Oie, "Forward error correction in ATM networks: an analysis of cell lossdistribution in a block," presented at INFOCOM'94. Networking for Global Communications., 13th Proceedings IEEE, 1994.

[8] S. Kang and D. Loguinov, "Impact of FEC overhead on scalable video streaming," presented at Proceedings of the international workshop on Network and operating systems support for digital audio and video, 2005.

[9] S. Yuk, M. Kang, D. Kim, B. Shin, and D. Cho, "Adaptive redundancy control for systematic erasure code based realtime data transmission in Internet," presented at 2000 IEEE International Conference on Communications, 2000. ICC 2000, 2000.

[10] S. Chin and R. Braun, "Improving video quality using packet interleaving, randomisationand redundancy," presented at Local Computer Networks, 2001. Proceedings. LCN 2001. 26th Annual IEEE Conference on, 2001.

[11] T. Stock and X. Garcia, "On the potentials of forward error correction mechanisms applied to real-time services carried over B-ISDN," Lecture Notes in Computer Science, vol. 1044, pp. 107-118, 1996

[12] R. Kalathur and M. Bassiouni, "Forward error correction with buffer management in multimedia ATMnetworks," presented at Southcon/94. Conference Record, 1994.

[13] X. Yu, J. Modestino, and Y. Chan, "A model-based evaluation methodology for assessing the efficacy of packet-level fec for delay-constrained video network transport," presented at 2006 IEEE International Conference on Image Processing, 2006.

[14] M. Gamage, M. Hayasaka, and T. Miki, "A connection-oriented network architecture with guaranteed QoS for future real-time applications over the Internet," *Computer Networks*, vol. 50, pp. 1130-1144, 2006.

[15] G. Dan, V. Fodor, and G. Karlsson, "A Rate-Distortion Based Comparison of Media-Dependent FEC and MDC for Real-Time Audio," presented at IEEE International Conference on Communications, 2006. ICC'06, 2006.

[16] F. Zhai, R. Beryy, T. Pappas, and A. Katsaggelos, "A rate-distortion optimized error control scheme for scalable video streaming over the Internet," presented at Multimedia and Expo, 2003. ICME'03. Proceedings. 2003 International Conference on, 2003.

[17] M. Johanson, "Adaptive forward error correction for real-time internet video," presented at Proc. of the 13th Packet Video Workshop, 2003.

[18] K. Park and W. Wang, "AFEC: an adaptive forward error correction protocol for end-to-endtransport of real-time traffic," presented at Computer Communications and Networks, 1998. Proceedings. 7th International Conference on, 1998.

[19] M. Hayasaka, M. Gamage, and T. Miki, "An efficient loss recovery scheme for on-demand video streaming over the internet," presented at Advanced Communication Technology, 2005, ICACT 2005. The 7th International Conference on.

[20] I. Cidon, A. Khamisy, and M. Sidi, "Analysis of packet loss processes in high-speed networks," *IEEE Transactions on Information Theory*, vol. 39, pp. 98-108, 1993.

[21] J. Bolot, T. Turletti, and S. INRIA, "Adaptive error control for packet video in the Internet," presented at Image Processing, 1996. Proceedings., International Conference on, 1996.

[22] X. Yu, J. Modestino, and I. Bajic, "Performance analysis of the efficacy of packet-level FEC in improving video transport over networks," presented at IEEE Interntional Conference on Image Processing, 2005. ICIP 2005, 2005.

[23] I. Rhee and S. Joshi, "Error recovery for interactive video transmission over the Internet," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 1033-1049, 2000

[24] R. Fang, D. Schonfeld, R. Ansari, and J. Leigh, "Forward error correction for multimedia and teleimmersion data streams," *FEC*, vol. 1, pp. P3.

[25] H. Wu, M. Claypool, and R. Kinicki, "Adjusting forward error correction with quality scaling for streaming MPEG," presented at Proceedings of the international workshop on Network and operating systems support for digital audio and video, 2005.

[26] A. Yamaguchi, M. Arai, and K. Iwasaki, "Evaluation of multicast error recovery using convolutional codes," presented at Dependable Computing, 2000. Proceedings. 2000 Pacific Rim International Symposium on, 2000.

[27] O. Almomani, S. Hassan and S. Awang Nor" Effects of Packet Size on FEC Performance "international conference on network Application, protocols and services 2008 (NetApps 2008), 21-22 november 2008

[28] O. Almomani, S. Hassan and O. Ghazali,"Optimalfec Blocksize Formedia Streaming Over Internet", in the Proceeding of the International Graduate Conference on Engineering and Science (IGCES 2008), Johor, Malaysia, 23rd-24th December 2008.

[29] N. Oguz, "A simulation study of forward error correction for lost packet recovery in B-ISDN/ATM," presented at 1993 IEEE International Symposium on Information Theory, 1993. Proceedings, 1993.

[30] K. Nahrstedt, "Quality of service in networked multimedia systems," *Handbook of Internet and Multimedia Systems and Applications*, pp. 217, 1999.

[31] R. Puri, K. Ramchandran, K. Lee, and V. Bharghavan, "Forward error correction (FEC) codes based multiple description coding for Internet video streaming and multicast," *Signal Processing: Image Communication*, vol. 16, pp. 745-762, 2001.

[32] J. Bolot, T. Turletti, and S. INRIA, "A rate control mechanism for packet video in the Internet," presented at INFOCOM'94. Networking for Global Communications., 13th Proceedings IEEE, 1994.

[33] D. Jurca and P. Frossard, "Optimal FEC Rate for Media Streaming in Active Networks," presented at Proceedings of IEEE ICME, 2004.

[34] A. Nafaa, T. Taleb, and L. Murphy, "Forward error correction strategies for media streaming over wireless networks," *IEEE Communications Magazine*, vol. 46, pp. 72, 2008.

[35] J. Ahn, Y. Lee, J. Yoon, and K. Lee, "Analyzing the Effect of a Block FEC Algorithm's Symbol Size on Energy Consumption in Wireless Sensor Networks," *Lecture Notes in Computer Science*, vol. 4239, pp. 440, 2006.

[36] W. Dapeng, Y. Hou, Z. Wen-wu, Z. Ya-Qin, and J. Peha, "Streaming video over the Internet: approaches and directions," *IEEE transactions on circuits and systems for video technology*, vol. 11, pp. 282-300, 2001.

[37] J. Boyce and R. Gaglianello, "Packet loss effects on MPEG video sent over the public Internet," presented at Proceedings of the sixth ACM international conference on Multimedia, 1998.

[38] X. Li, M. Ammar, and S. Paul, "Video multicast over the Internet," *IEEE network*, vol. 13, pp. 46-60, 1999.

[39] E. Lin, C. Podilchuk, T. Kalker, and E. Delp, "Streaming video and rate scalable compression: What are the challenges for watermarking?," *Journal of Electronic Imaging*, vol. 13, pp. 198, 2004..

[40] D. Wu, Y. Hou, and Y. Zhang, "Transporting real-time video over the Internet: challenges and approaches," *Proceedings of the IEEE*, vol. 88, pp. 1855-1877, 2000.

[41] Q. Zhao, C. Li, X. Zhang, and D. A.-L. Liu, Dongtao, "ANew FEC Scheme for Real-time Transmission of High Definition Video over IP Network," presented at Intelligent Pervasive Computing, 2007. IPC. The 2007International Conference on, 2007.

[42] S.-R. Kang and D. Loguinov, "Modeling Best-Effort and FEC Streaming of Scalable Video in Lossy Network Channels," Networking, IEEE/ACM Transactions on, vol. 15, pp. 187-200, 2007.

[43] K. Munadi, M. Kurosaki, K. Nishikawa, and H. Kiya, "Efficient packet loss protection for JPEG2000 images enabling backward compatibility with a standard decoder," presented at Image Processing, 2004. ICIP'04. 2004 International Conference on, 2004.

[44] G. Dán, V. Fodor, and G. Karlsson, "On the effects of the packet size distribution on FEC performance," *Computer Networks*, vol. 50, pp. 1104-1129, 2006.

[45] H. Matsuoka, T. Yoshimura, and T. Ohya, "End-to-end robust IP soft handover," presented at IEEE International Conference on Communications, 2003. ICC'03, 2003.

[46] S. Aoki, T. Shimizu, S. Nishimura, and K. Aoki, "Efficient Reliable Multicast Using FEC and Limited Retransmission for HDTV IP Broadcasting," presented at 5th IEEE Consumer Communications and Networking Conference, 2008. CCNC 2008, 2008.

[47] T. Kondo, K. Nishimura, and R. Aibara, "Implementation and evaluation of the robust high-quality video transfer system on the broadband Internet," presented at Applications and the Internet, 2004. Proceedings. 2004 International Symposium on, 2004

[48] F. Zhai, R. Beryy, T. Pappas, and A. Katsaggelos, "A rate-distortion optimized error control scheme for scalable video streaming over the Internet," presented at Multimedia and Expo, 2003. ICME'03. Proceedings. 2003 International Conference on, 2003.

[49] S. Jharee and K. Soyjaudah, "Recovery of small bursts of voice packets using RS erasure code with low decoding delay," *AFRICON 2007*, pp. 1-7, 2007.

[50] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman, "Efficient erasure correcting codes," *IEEE Transactions on Information Theory*, vol. 47, pp. 569-584, 2001.

[51] J. Park, E. Chong, and H. Siegel, "Efficient multicast stream authentication using erasure codes," *ACM Transactions on Information and System Security (TISSEC)*, vol. 6, pp. 258-285, 2003.

[52] R. Morelos-Zaragoza, J. Wiley, and I. Sons, *The art of error correcting coding*: Wiley, 2006. , Ltd. ISBN: 0-470-01558-6

[53] T.Issariyakul and E. Hossain, "Introduction to NetworkSimulator NS2", ISBN: 978-0-387-71759-3 e-ISBN: 978-0-387-71760-9, DOI: 10.1007/978-0-387-71760-9, Library of Congress Control Number: 2008928147, 2009 Springer Science+Business Media, LLC

[54] E. Biersack, "A simulation study of forward error correction in ATM networks."

[55] N. Oguz and E. Ayanoglu, "A simulation study of two-level forward error correction for lost packet recovery in B-ISDN/ATM," presented at IEEE International Conference on Communications, 1993. ICC 93. Geneva. Technical Program, Conference Record, 1993.

[56] O. Ghazali, "scalable and smooth TCP-friendly received-based layered multicast protocol,", Ph.D. dissertation, Malaysia University Utara Malaysia,2008

[57] P. Frossard, "FEC performance in multimedia streaming," *IEEE Communications Letters*, vol. 5, pp. 122-124, 2001.

[58] K. Harfoush, A. Bestavros, and J. Byers, "Measuring bottleneck bandwidth of targeted path segments," presented at IEEE INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies.

[59] W. Alyson and L. Nikolaos ,"Modern Statistical and Mathematical Methods In Reliability", LC Call Number: TA169 -- .M64 2005eb,ISBN: 9789812563569 9789812703378 ,Dewey Decimal Number: 620/.00452,2005

Appendices

Appendix A

The mean of lost and received packets for each run

1) Parity Amount=10 packets		2) Parity Amount=15 packets	
Loss packet	Recived	Loss packet	Recived packet
834	5466	852	5598
836	5464	857	5593
855	5445	863	5587
830	5470	856	5594
837	5463	865	5585
861	5439	862	5588
841	5459	881	5569
835	5465	851	5599
833	5467	853	5597
835	5465	878	5572
832	5468	862	5588
844	5456	858	5592
831	5469	858	5592
847	5453	858	5592
848	5452	844	5606
835	5465	854	5596
839	5461	855	5595
839	5461	865	5585
859	5441	862	5588
827	5473	855	5595
839.9	5460.1	859.45	5590.55

3) Parity Amount=20 packets

Loss packet	Recived
867	5733
878	5722
884	5716
869	5731
886	5714
881	5719
881	5719
901	5699
899	5701
859	5741
881	5719
897	5703
879	5721
865	5735
888	5712
883	5717
880	5720
881	5719
875	5725
875	5725
880.45	5719.55

4) Parity Amount=25 packets

Loss packet	Recived packet
917	5833
898	5852
903	5847
893	5857
898	5852
894	5856
911	5839
897	5853
891	5859
905	5845
899	5851
893	5857
909	5841
898	5852
889	5861
920	5830
898	5852
888	5862
903	5847
897	5853
900.05	5849.95

5) Parity Amount=30 packets

Loss packet	Recived packet
936	5964
919	5981
902	5998
904	5996
913	5987
911	5989
912	5988
922	5978
947	5953
913	5987
910	5990
922	5978
915	5985
914	5986
918	5982
976	5924
919	5981
936	5964
938	5962
912	5988
921.95	5978.05

6) Parity Amount=35 packets

Loss packet	Recived packet
930	6120
946	6104
952	6098
939	6111
934	6116
927	6123
946	6104
944	6106
930	6120
951	6099
930	6120
943	6107
939	6111
928	6122
938	6112
932	6118
927	6123
943	6107
931	6119
925	6125
936.75	6113.25

Appendix B

The mean of lost and received packets for each run, and PLR after and before using FEC

1) Parity amount = 10 packets

Loss packet	Received packet	PLR before using FEC	PLR after using FEC
834	5466	0.139	0.132380952
836	5464	0.139333333	0.132698413
855	5445	0.1425	0.135714286
830	5470	0.138333333	0.131746032
837	5463	0.1395	0.132857143
861	5439	0.1435	0.136666667
841	5459	0.140166667	0.133492063
835	5465	0.139166667	0.132539683
833	5467	0.138833333	0.132222222
835	5465	0.139166667	0.132539683
832	5468	0.138666667	0.132063492
844	5456	0.140666667	0.133968254
831	5469	0.1385	0.131904762
847	5453	0.141166667	0.134444444
848	5452	0.141333333	0.134603175
835	5465	0.139166667	0.132539683
839	5461	0.139833333	0.133174603
839	5461	0.139833333	0.133174603
859	5441	0.143166667	0.136349206
827	5473	0.137833333	0.131269841
839.9	5460.1	0.139983333	0.13331746

2) Parity amount = 15 packets

Loss packet	Received packet	PLR before using FEC	PLR After using FEC
852	5598	0.142	0.132093023
857	5593	0.142833333	0.132868217
863	5587	0.143833333	0.13379845
856	5594	0.142666667	0.132713178
865	5585	0.144166667	0.134108527
862	5588	0.143666667	0.133643411
881	5569	0.146833333	0.136589147
851	5599	0.141833333	0.131937984
853	5597	0.142166667	0.132248062
878	5572	0.146333333	0.136124031
862	5588	0.143666667	0.133643411
858	5592	0.143	0.133023256
858	5592	0.143	0.133023256
858	5592	0.143	0.133023256
844	5606	0.140666667	0.130852713
854	5596	0.142333333	0.132403101
855	5595	0.1425	0.13255814
865	5585	0.144166667	0.134108527
862	5588	0.143666667	0.133643411
855	5595	0.1425	0.13255814
859.45	5590.55	0.143241667	0.133248062

3) Parity amount = 20 packets

Loss packet	Received packet	PLR before using FEC	PLR after using FEC
867	5733	0.1445	0.131363636
878	5722	0.14633333	0.133030303
884	5716	0.14733333	0.133939394
869	5731	0.14483333	0.131666667
886	5714	0.14766667	0.134242424
881	5719	0.14683333	0.133484848
881	5719	0.14683333	0.133484848
901	5699	0.15016667	0.136515152
899	5701	0.14983333	0.136212121
859	5741	0.14316667	0.130151515
881	5719	0.14683333	0.133484848
897	5703	0.1495	0.135909091
879	5721	0.1465	0.133181818
865	5735	0.14416667	0.131060606
888	5712	0.148	0.134545455
883	5717	0.14716667	0.133787879
880	5720	0.14666667	0.133333333
881	5719	0.14683333	0.133484848
875	5725	0.14583333	0.132575758
875	5725	0.14583333	0.132575758
880.45	5719.55	0.14674167	0.133401515

4) Parity amount = 25 packets

Loss packet	Received packet	PLR before using FEC	PLR after using FEC
917	5833	0.152833333	0.135851852
898	5852	0.149666667	0.133037037
903	5847	0.1505	0.133777778
893	5857	0.148833333	0.132296296
898	5852	0.149666667	0.133037037
894	5856	0.149	0.132444444
911	5839	0.151833333	0.134962963
897	5853	0.1495	0.132888889
891	5859	0.1485	0.132
905	5845	0.150833333	0.134074074
899	5851	0.149833333	0.133185185
893	5857	0.148833333	0.132296296
909	5841	0.1515	0.134666667
898	5852	0.149666667	0.133037037
889	5861	0.148166667	0.131703704
920	5830	0.153333333	0.136296296
898	5852	0.149666667	0.133037037
888	5862	0.148	0.131555556
903	5847	0.1505	0.133777778
897	5853	0.1495	0.132888889
900.05	5849.95	0.150008333	0.133340741

5) Parity amount = 30 packets

Loss packet	Received packet	PLR before using FEC	PLR After using FEC
936	5964	0.156	0.135652174
919	5981	0.153166667	0.133188406
902	5998	0.150333333	0.130724638
904	5996	0.150666667	0.131014493
913	5987	0.152166667	0.132318841
911	5989	0.151833333	0.132028986
912	5988	0.152	0.132173913
922	5978	0.153666667	0.133623188
947	5953	0.157833333	0.137246377
913	5987	0.152166667	0.132318841
910	5990	0.151666667	0.131884058
922	5978	0.153666667	0.133623188
915	5985	0.1525	0.132608696
914	5986	0.152333333	0.132463768
918	5982	0.153	0.133043478
976	5924	0.162666667	0.141449275
919	5981	0.153166667	0.133188406
936	5964	0.156	0.135652174
938	5962	0.156333333	0.135942029
912	5988	0.152	0.132173913
921.95	5978.05	0.153658333	0.133615942

6) **Parity amount = 35 packets**

Loss packet	Received packet	PLR before using FEC	PLR After using FEC
930	6120	0.155	0.13191489
946	6104	0.157666667	0.1341844
952	6098	0.158666667	0.13503546
939	6111	0.1565	0.13319149
934	6116	0.155666667	0.13248227
927	6123	0.1545	0.13148936
946	6104	0.157666667	0.1341844
944	6106	0.157333333	0.13390071
930	6120	0.155	0.13191489
951	6099	0.1585	0.13489362
930	6120	0.155	0.13191489
943	6107	0.157166667	0.13375887
939	6111	0.1565	0.13319149
928	6122	0.154666667	0.13163121
938	6112	0.156333333	0.13304965
932	6118	0.155333333	0.13219858
927	6123	0.1545	0.13148936
943	6107	0.157166667	0.13375887
931	6119	0.155166667	0.13205674
925	6125	0.154166667	0.13120567
936.75	6113.25	0.156125	0.13287234