ARMOR - Adjusting Repair and Media Scaling with Operations Research for Streaming Video

by

Huahui Wu

A Dissertation
Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment
of the Requirements for the
Degree of Doctor of Philosophy

in

Computer Science

by

May 2006

APPROVED:

Prof. Mark Claypool Advisor	Prof. Robert Kinicki Co-Advisor
Prof. Craig Wills Committee Member	Prof. Wu-chi Feng External Committee Member Computer Science, Portland State University
Prof. Michael Gennert Head of Department	

Abstract

Streaming multimedia quality is impacted by two main factors: capacity constraint and packet loss. To match the capacity constraint while preserving real-time playout, media scaling can be used to discard the encoded multimedia content that has the least impact on perceived video quality. To limit the impact of lost packets, repair techniques, e.g. forward error correction (FEC), can be used to repair frames damaged by packet loss. However, adding data to facilitate repair requires further reduction of the original multimedia data, making the decision of how much repair data to use of critical importance. Assuming a limited network capacity and the availability of an estimate of the current packet loss rate along a flow path, selecting the best distribution of FEC packets for video frames with inherent interframe encoding dependencies can be cast as a constraint optimization problem that attempts to optimize the quality of the video stream.

This thesis presents an *Adjusting Repair and Media scaling with Operations Research (ARMOR)* system. An analytical model is derived for streaming video with FEC and media scaling. Given parameters to represent network loss as well as video frame types and sizes, if the number of FEC packets per video frame type and media scaling pattern is specified, the model can estimate the video quality at the receiver side. The model is then used in an operations research algorithm to adjust the FEC strength and media scaling level to yield the best quality under the capacity constraint. Four different combinations of FEC type and media scaling method are studied: Media Independent FEC with Temporal Scaling (MITS), Media Independent FEC with Quality Scaling (MIQS), Media Independent FEC with Temporal and Quality Scaling

(MITQS), and Media Dependent FEC with Quality Scaling (MDQS). The analytical experiments show: 1) adjusting FEC always achieves a higher video quality than streaming video without FEC or with a fixed amount of FEC; 2) Quality Scaling usually works better than Temporal Scaling; and 3) Media Dependent FEC (MDFEC) is typically less effective than Media Independent FEC (MIFEC). A user study is presented with results from 74 participants analysis shows that the ARMOR model can accurately estimate users' perceptual quality. Well-designed simulations and a realistic system implementation suggests the ARMOR system can practically improve the quality of streaming video.

Acknowledgments

Thank my parents! They brought me to this wonderful world and support everything I did, am doing, and will do.

Thank Professor Claypool and Professor Kinicki! They advise me with the right direction and the right way to do research, and they also give me enough space to think and try.

Thank Mark and Bob again! They care about my personal life and help me to grow up, as Jae said, we are like a family.

Thank Professor Wills! Our offices are so close and he can keep eyes on me to make sure I am working hard in my office.

Thank Professor Feng at Portland State University! It's valuable to have him in my committee and thank for attending my proposal presentation and my dissertation defense from a city far far away.

Thank Professor Gennert! His "Computer Vision" class is one of my favorite classes. Sorry about breaking his 4x4x4 magic cube.

Thank Jae Chung, Feng Li, Mingzhe Li, Rui Lu, Chunling Ma and Hao Shang! They give me all sorts of supports from a candy to a dinner.

Thank my wife, Chunfang! She supports me on everything and she even canceled her weekly shopping for two months because of this dissertation!

Thank Marvin! He reads it many times everyday and, occasionally, he "points out" some of my mistakes, but most amazingly, he is a cat!

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 The Dissertation 5

1.3 Contribution 10

1.4 Road map 12

2 Background **13**

2.1 Video Compression Standard 13

2.1.1 MPEG 13

2.1.2 H.26X 16

2.2 Forward Error Correction 17

2.2.1 Media-Independent FEC (MIFEC) 17

2.2.2 Media-Dependent FEC (MDFEC) 18

2.3 Capacity Constraint 20

2.3.1 TCP-Friendly Flow 20

2.3.2 Maximum Available Bandwidth 21

2.4 Media Scaling 22

2.4.1 Post-Encoding Temporal Scaling (POTS) 23

2.4.2 Pre-Encoding Temporal Scaling (PETS) 24

2.4.3 PETS vs. POTS 26

2.4.4 Quality Scaling 28
2.5 Quality Measurement 30
2.5.1 Network Goodput 30
2.5.2 Playable Frame Rate 31
2.5.3 Peak Signal Noise Ratio (PSNR) 31
2.5.4 Video Quality Metric (VQM) 32
2.5.5 Subjective Measurement 33

3 Related Work **36**
3.1 Media Repair 36
3.1.1 Media Repair Taxonomy 36
3.1.2 Retransmission 37
3.1.3 Interleaving 38
3.1.4 Media Independent FEC 39
3.1.5 Media Dependent FEC 40
3.1.6 Combinations 41
3.1.7 Our approach 41
3.2 Media Scaling 42
3.2.1 Media Scaling in Research 43
3.2.2 Media Scaling in Commercial Software 44
3.2.3 Our approach 44
3.3 Media Repair with Scaling 45

4 Analytical Models and Optimization Algorithms **47**
4.1 Media-Independent FEC with Temporal Scaling (MITS) . . . 49
4.1.1 System Layers 49
4.1.2 Successful Frame Transmission Probabilities 50
4.1.3 Capacity Constraint 51

4.1.4 Media-Independent FEC with POst-encoded Temporal Scaling (MIPOTS) . . . 51
4.1.5 Media-Independent FEC with Pre-Encoded Temporal Scaling (MIPETS) . . . 55
4.1.6 Summary . . . 61
4.2 Study of GOP Length . . . 62
4.2.1 Overview . . . 62
4.2.2 Static MPEG Files . . . 63
4.2.3 Streaming MPEG . . . 71
4.2.4 Analysis . . . 72
4.2.5 Summary . . . 75
4.3 Analytical Experiments of MITS . . . 77
4.3.1 Analytical Experiments of MIPOTS . . . 77
4.3.2 Analytical Experiments of MIPETS . . . 84
4.3.3 Summary . . . 89
4.4 Media-Independent FEC and Quality Scaling (MIQS) . . . 90
4.4.1 System Layers . . . 90
4.4.2 Distortion from Quality Scaling . . . 91
4.4.3 Playable Frame Rate . . . 92
4.4.4 Distorted Playable Frame Rate . . . 93
4.4.5 Optimization Algorithm . . . 95
4.4.6 Analytical Experiments . . . 96
4.4.7 Summary . . . 102
4.5 Media Independent FEC with Temporal and Quality Scaling (MITQS) . . . 104
4.5.1 Background . . . 105
4.5.2 System Layers . . . 106

4.5.3 Distortion from Quality Scaling . . . 107
4.5.4 Playable Frame Rate . . . 107
4.5.5 Distorted Playable Frame Rate . . . 109
4.5.6 Optimization Algorithm . . . 109
4.5.7 Analytical Experiments . . . 111
4.5.8 Summary . . . 114
4.6 Media Dependent FEC and Quality Scaling (MDQS) . . . 116
4.6.1 System Introduction . . . 116
4.6.2 Frame Size . . . 117
4.6.3 Successful Frame Transmission Probabilities . . . 117
4.6.4 Weighted Playable Frame Rate . . . 118
4.6.5 Optimal Weighted Playable Frame Rate . . . 121
4.6.6 Revisiting of Media Independent FEC with Quality Scaling (MIQS) . . . 122
4.6.7 Comparison of MDQS and MIQS . . . 124
4.6.8 Summary . . . 126

5 User Study 128

5.1 Methodology . . . 129
5.2 User Study Application . . . 130
5.3 Video Clips . . . 133
5.4 User Demographics . . . 134
5.5 Results . . . 136
5.5.1 Video Quality Measurements . . . 136
5.5.2 ARMOR Layer Variables . . . 140
5.5.3 MPEG and Network Parameters . . . 142
5.6 Summary . . . 145

6 Simulations **147**

6.1 Inaccurate Loss Prediction 149

6.2 Bursty Loss 150

6.3 Variable Round-Trip Times 152

6.4 Variable MPEG Frame Sizes 153

6.5 Combination Effects 156

6.6 Practical Considerations 158

7 Implementation **160**

7.1 Background 160

7.2 System Modules 162

7.3 System Testing and Evaluation 166

7.3.1 System Settings 166

7.3.2 Loss Rate Prediction 166

7.3.3 TCP-Friendly Behavior 168

7.3.4 System Performance Evaluation 170

7.3.5 Comparison of FEC schemes 172

7.4 Summary 174

8 Conclusions **176**

8.1 Analytical Models and Optimization Algorithms 177

8.1.1 Media Independent FEC with Temporal Scaling (MITS) 177

8.1.2 Media Independent FEC with Quality Scaling (MIQS) 178

8.1.3 Media Independent FEC with Temporal and Quality Scaling (MITQS) 179

8.1.4 Media Dependent FEC with Quality Scaling (MDQS) . 180

8.2 User Study 181

8.3 Simulation 181

8.4 Implementation 182

9 Future Work **184**

List of Figures

1.1 An General ARMOR System Architecture 6

2.1 A sample MPEG Group Of Pictures (GOP) 15
2.2 Reed-Solomon code 18
2.3 Media-Dependent FEC 19
2.4 Discarding frames in POTS 23
2.5 Discarding frames in PETS 25
2.6 MPEG encoding (from [30, 54]) 28

3.1 Taxonomy of media repair (from [64]) 37

4.1 Screenshots of Video Clips 64
4.2 Impact of N_{BP} on MPEG files for the other videos 68
4.3 Impact of N_P on MPEG files for the other videos 70
4.4 Streaming *Foreman* with FEC and PETS. Network model has 2% loss and 1.5 Mbps capacity constraint 73
4.5 Streaming the other 8 videos with adjusted FEC and PETS, 2% loss and 1.5 Mbps capacity constraint ($N_{BP} = 2$). 74
4.6 Corresponding δ values in Figure 4.5 75
4.7 Comparison of Playable Frame Rates 81
4.8 Adjusted FEC Pattern 82
4.9 Comparison of Playable Frame Rates 86

4.10 Adjusted FEC Pattern 87
4.11 Temporal Scaling Level 88
4.12 Distorted Playable Frame Rates 100
4.13 Comparison of scaling methods, with the 1st row of low motion clips, the 2nd row of medium motion clips and the 3rd row of high motion clips 112
4.14 Comparison of FEC methods, with the 1st row of low motion clips, the 2nd row of medium motion clips and the 3rd row of high motion clips 114
4.15 Comparison of MDQS and MIQS 126

5.1 Screenshot of the User Information Dialog 131
5.2 Screenshot of the Directions 132
5.3 Screenshot of the Main Dialog, where users watch a pair of video clips and compare the second clip to the first clip. 132
5.4 User Demographics 135
5.5 R_D vs. User Score 136
5.6 R vs. User Score 137
5.7 VQM Distortion vs. User Score 138
5.8 PSNR vs. User Score 139
5.9 Adjusted FEC vs. Non-FEC 141
5.10 Temporal Scaling vs. Quality Scaling 142
5.11 Low Motion Video *News* vs. High Motion Video *Coastguard* . 143
5.12 1% Loss vs. 4% Loss 144

6.1 Impact of Inaccurate Loss Prediction 149
6.2 Impact of Bursty Loss 151
6.3 Impact of Variable RTT 152

6.4 Impact of Variable Frame Size for MIPOTS 154
6.5 Impact of Variable Frame Size for MIQS 156
6.6 Combination Effects 157

7.1 ARMOR System 162
7.2 ARMOR System Data Formats 165
7.3 Loss Rate Prediction 168
7.4 Throughput of ARMOR Traffic and Wget Traffic with 2% Loss 169
7.5 Performance of ARMOR System 172
7.6 Comparison of FEC methods in Realistic ARMOR System . . 174

List of Tables

2.1 Some ISP plans 22

2.2 Temporal Scaling Characteristics 25

2.3 PETS vs. POTS 26

4.1 System Layers and Parameters/Variables 50

4.2 Video Clips 65

4.3 Impact of N_{BP} on MPEG files for *Foreman* 66

4.4 Impact of N_P on MPEG files for *Foreman* 69

4.5 System Parameter Settings 80

4.6 Temporal Scaling Patterns 84

4.7 System Layers and Parameters/Variables 90

4.8 System Parameter Settings 97

4.9 Estimated Value by Equation 4.32 versus Real Video Data . . 99

4.10 Preliminary User Study 101

4.11 Quality Scaling Levels 105

4.12 Temporal Scaling Levels 106

4.13 System Layers and Parameters/Variables 106

4.14 System Layers and Parameters/Variables 116

4.15 System Parameter Settings 125

5.1 Clip factor combination 134

7.1 System Parameter Settings 166
7.2 Throughput of ARMOR and Wget with 2% Loss for Five Runs 169

Chapter 1

Introduction

1.1 Motivation

As the number of active Internet users continues to grow and streaming media applications become more commonplace, the number of users and volume of data on the Internet is increasing at an explosive rate. The sheer number of possible users and applications at any point in time raises the probability of streaming multimedia flows encountering bandwidth constraints due to congestion or otherwise limited capacity. Although the bandwidth to the last-mile hop has increased rapidly in the past few years, high quality video streaming can still expand to use even more bandwidth than the network can support. In particular, streaming video applications are being to be introduced into the cellular phone market, where available bandwidth can severely limit the streaming video quality.

In managing congestion, the Transmission Control Protocol (TCP) is the de facto standard transport protocol for typical Internet applications. To overcome short-term congestion and avoid long-term congestion collapse, TCP uses additive increase and multiplicative decrease (AIMD) and other conges-

tion control mechanisms. However, streaming media prefers a steady data rate rather than the bursty data rate often associated with TCP. Besides, unlike traditional Internet applications, streaming video is sensitive to delay and jitter, but can tolerate some data loss. Since TCP reacts to packet loss through retransmission, this introduces latency to the transmission, and hence, streaming video applications often use UDP as their transport protocol rather than TCP.

While streaming flows have traditionally selected UDP over TCP [52, 83], there is a growing consensus that all Internet applications must be TCP-Friendly. A flow is *TCP-Friendly* if its data rate does not exceed the maximum data rate of a conformant TCP connection under equivalent network conditions. There are proposed approaches to detect and restrict the capacity of non-TCP friendly flows [49]. Thus, networking researchers have proposed new TCP-Friendly protocols (e.g. TFRC) [3, 26, 74] for transporting streaming media. Moreover, if streaming media applications use TCP-Friendly streaming protocols, the network congestion control techniques can more effectively respond to all forms of congestion. This, in turn, should yield better overall quality of service for streaming flows.

Another capacity constraint comes from the restricted capacity in the "last mile" hop to end users. For example, a typical DSL company in Worcester area offers 1.5 Mbps home user speed. From the view point of the end-user, the video-streaming flow may not need to be TCP-Friendly, and it can use all of the capacity allocated to the end-user, or at least more than a TCP-Friendly flow. However, the capacity is still limited by the final link's capacity. Major commercial streaming software has been implemented to expect such a constraint, such as Microsoft's Windows Media Player [58].

To preserve real-time streaming media playout, streaming servers must

scale back their streaming data rate to match the TCP-Friendly data rate or the capacity constraint. This proactive data rate reduction by the multimedia server is called *media scaling* [6, 82]. Armed with knowledge about the relative importance of specific frame types and interframe dependencies, a multimedia application can discard the least significant packets with respect to perceived quality.

There are two typical ways of doing media scaling:

1. *Temporal Scaling* reduces bitrates by discarding frames before transmission. Temporal scaling can be further classified into two approaches: In *Post-encoding Temporal Scaling (POTS)*, the scaling is processed after encoding where the system discards the encoded B or P frames before sending them to the network. In *Pre-encoding Temporal Scaling (PETS)*, the scaling is processed before encoding where the system discards the raw images before encoding them into video frames.

2. *Quality scaling* reduces bitrates by reducing visual quality and details. For example, in MPEG, each frame is divided into 8x8 pixel macroblocks, which are converted to 64 coefficients after the discrete-cosine transform (DCT). Then, the encoder uses quantization to remove the low order bits from these coefficients to get compression benefits. By using a higher quantization level, more coefficients are removed and a lower bitrate is achieved at a cost of reduced visual quality.

While multimedia applications can tolerate some data loss, excessive packet loss during congestion yields unacceptable media quality. Explicit Congestion Notification (ECN) [24] can reduce packet loss dramatically by marking packets instead of dropping them. Unfortunately, ECN has not been and is not likely to be widely deployed [18]. Moreover, in wireless networks, packet loss

from the physical layer are not typically indications of congestion and can not be reduced by ECN. Since video encoding involves interframe dependencies [54], the random dropping of packets by routers can seriously degrade video quality. In MPEG, for example, dropping packets from an independently encoded I frame causes the following dependent P and B frames to not be fully decodable. In practice, interframe dependencies have been shown to cause a 3% packet loss rate to result in a 30% frame loss rate [10].

While packet losses can be repaired by retransmission, applications such as videoconferencing and interactive virtual reality application cannot afford the increased latency required for retransmission, especially for connections with high round-trip times. Although backbone routers reduce queuing delays by bandwidth over-provisioning, typical end-to-end round-trip times are still much higher than typical interactive video latency requirements. In a measurement study of Internet streaming video [15], the median RTTs for different capacity configurations are all over 100ms. This suggests that utilizing lower latency repair approaches, such as Forward Error Correction (FEC), in conjunction with TCP-Friendly protocols to deliver streaming applications over the Internet is important. Used properly, FEC [7, 57, 60, 62] can reduce or eliminate the impact from packet loss and partially or fully insulate video applications from degraded quality [47].

There are two typical methods for doing FEC:

1. *Media-Independent FEC* does not rely upon knowledge of the content, and instead uses a mathematic algorithm to generate redundant parity bits from the original data. When there are losses, the correctly received data and the parities can be used to reconstruct the original data.

2. *Media-Dependent FEC* uses knowledge of the content, adding lower quality data to the original data. When there is a loss, the original data can

be replaced with the lower quality data.

However, both FEC types require additional repair data to be added to the original video data. If a streaming video is to operate within a capacity limit, the additional FEC data will reduce the effective transmission rate of the original video content. In this thesis, determining the optimal combination of FEC, which repairs the packet loss, and media scaling, which reduces transmission bitrate, is addressed for video streaming.

1.2 The Dissertation

Assuming a limited network capacity and the availability of an estimate of the current packet loss rate along a flow path, selecting the best distribution of FEC packets for video frames with inherent interframe encoding dependencies can be cast as a constraint optimization problem that attempts to optimize the quality of the video stream [50]. Current approaches use either presumptive, static FEC choices [1, 34] or adapt FEC to perceived packet loss on the network without regard to data rate constraints [7, 60, 62].

In this project, building upon the work of Mayer-Patel *et al.* [50], an analytical model is derived for streaming video with FEC [85], where media scaling is used to keep the data rate under the constraint. Given parameters to represent network loss and video frame types and sizes, if the number of FEC packets per video frame type and media scaling pattern is specified, the model can estimate the video quality at the receiver side. For different media scaling types, different quality measurements are used. The model is then used in an operations research algorithm to exhaustively search all possible combinations of FEC and media scaling patterns to find the combination of FEC and media scaling that yields the best quality under the capacity constraint.

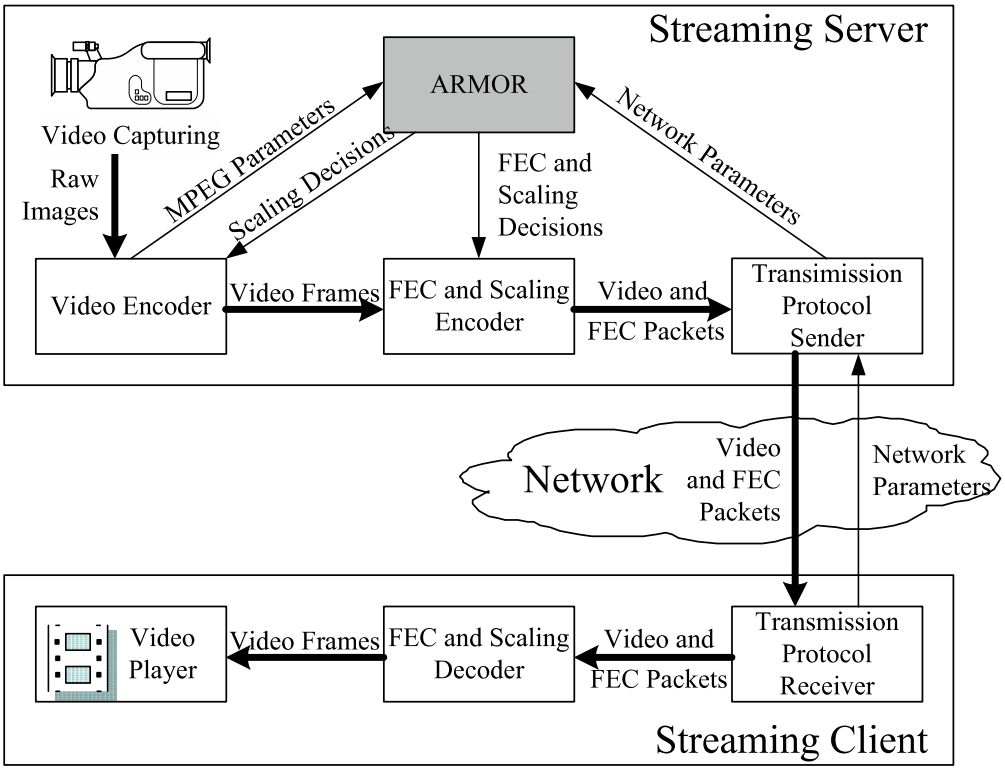


Figure 1.1: An General ARMOR System Architecture

The combination of the video quality model and the optimization algorithm is named as *Adjusting Repair and Media scaling with Operations Research (ARMOR)*. ARMOR can be used in a video streaming system as a control module. The general architecture of an ARMOR system[1] is depicted by Figure 1.1, with slight variances of the system for different types of FEC and media scaling.

At the sender side, assuming the raw images are provided by the video capture device, the raw images are then encoded into video frames by the encoder. The FEC encoder takes the video frames from the encoder and adds redundancy into each frame, and then sends the video frames with FEC to the transmission protocol, where the frames are split into packets and sent

[1]We define "ARMOR system" as a streaming system which uses ARMOR to adjust the repair and media scaling.

to the network. The ARMOR module works in conjunction with these modules. It takes the parameters of the video stream such as the GOP pattern and frame sizes, as well as the parameters from the transmission protocol and determines the best encoding, scaling methods and FEC amount to produce the optimal quality. The ARMOR module then selects the optimal FEC and scaling combination and feeds this information to the video encoder and FEC encoder. At the receiver side, the transmission protocol receives the packets and passes them to the FEC decoder. It also determines network characteristics such packet loss probability and round-trip time and informs the sender periodically. The FEC decoder then recovers the original video frames from the video packets plus FEC packets and sends the video frames to the player, where the video frames are played out.

In this thesis, different combinations of media scaling and forward error correction are studied. Specifically, they are:

1. Media Independent FEC with Temporal Scaling (MITS), which discards frames during media scaling and adds redundant packets for repairing;

2. Media Independent FEC with Quality Scaling (MIQS), which reduces video quality during media scaling and adds redundant packets for repairing;

3. Media Independent FEC with Temporal and Quality Scaling (MITQS), which discards frames and reduces video quality during media scaling and adds redundant packets for repairing;

4. Media Dependent FEC with Quality Scaling (MDQS), which reduces quality during media scaling and adds lower quality frames for repairing.

We also planned to study Media Dependent FEC with Temporal Scaling (MDTS), which discards frames for media scaling and adds lower quality video

frames for repairing. However, our results of MDQS show Media Dependent FEC is less effective than Media Independent FEC, so the thesis discontinue the investigation of MDTS and focus more on Media Independent FEC.

The analytical calculations required by the model and optimization algorithm are expected to be done in real-time, making the determination of optimal choices for adaptive FEC feasible for most streaming multimedia connections. Moreover, all the calculation could be processed before transmission and the results could be put into a hash table. During transmission, the optimal choice could just be determined from the table.

After studying ARMOR with analytical experiments, the ARMOR model is tested with a comprehensive user study to validate if video quality estimated by the model is correlated to the user's perceptual quality. Individual users are asked to view 16 pairs of video clips and rate the quality differences for each pair. In each pair, the first clip has the original quality before network transmission and the second clip is transmitted with a FEC and scaling pattern. The results are then analyzed to check if the model can accurately capture the quality distortion of streaming video.

It is assumed that the system can accurately estimate packet loss probability, round-trip time and video frame information. Practically, it is necessary to validate ARMOR and measure the effectiveness by simulating the ARMOR model and algorithm with realistic Internet conditions, where the prediction of network information may have errors. Also, since the analytical system assumes constant round-trip time and fixed video frame sizes, additional simulation experiments are designed with trace-driven round-trip times and real video frame sizes to validate the system. Moreover, the implementation on a real network is designed to validate and evaluate the system and analyze other realistic measurements. The cumulative effect of these experiments lends cred-

ibility to the fact that using the ARMOR model and algorithm to adjust FEC with media scaling can be effectively used to provide high quality streaming video.

In the ARMOR optimization algorithm, the search space is restricted by several assumptions. One assumption is that the Group of Pictures (GOP) length is not large, namely, no bigger than 15. There are a couple of reasons for this assumption, such as a short GOP is good for VCR-like functions, has fewer propagation errors, and uses less time in the algorithm. A study of the GOP length is necessary to validate some of these assumptions. Experiments are built to study how a longer GOP increases the average frame size and how it impacts visual quality. The results show: 1) the number of B frames between two reference frames should be set to two when the video stream does not have severe delay constraints, and 2) the number of P frames should be 5 or fewer as there is little performance gain in setting the number of P frames in the GOP larger than 5.

Audio streaming is not studied in this thesis, but it can be continued in a manner similar to the approach of Media Dependent FEC with Quality Scaling. First, typical audio frames do not have the same temporal dependencies as video frame, so every audio frame can be modeled as is an MPEG I frame. Second, since quality scaling and media-dependent FEC are widely used in audio streaming, the MDQS system could be studied with audio as in "Section 4.6 MDQS". Third, there are lots of quality measurement tools for audio, such as Signal-to-Noise Ratio (SNR), Perceptual Speech Quality Measurement (PSQM) [4], Perceptual Analysis Measurement System (PAMS) [76] and the ITU E-model [29]. Since the E-model is recommended because of its high correlation to the subjective measurement [29], it could be used as the measure of performance when evaluating different scaling and FEC approaches. Finally,

the qualities of audio and video would need to be weighted. The overall quality would be the optimized under the capacity constraint. User studies could be conducted to test the system's performance.

Notice, the target application of this work could be video conferencing where FEC is used as a low-latency repair technique. Even though this thesis studies a wide range of video content from low to high motion, typical video conferencing sessions may only have low motion content. This constraint could allow the ARMOR adaptions to be tuned for greater effectiveness, and is left as future work.

1.3 Contribution

The main contributions of this dissertation are the design, user study, simulation, and implementation of the *Adjusting Repair and Media scaling with Operations Research (ARMOR)* systems. The specific contributions of the dissertation include:

1. The playable probabilities of video frames are captured analytically. The successful transmission probability of each frame is computed based on the frame size and the FEC redundancy. The dependencies among frame types are used to determine if the received frames are playable. Then the total playable frame rate is added up as a measure of streaming quality after distortion from packet loss and temporal scaling.

2. The quantization distortion of streaming video, which is caused by low accuracy of the DCT coefficients and appears as coarse granularity in every frame, is modeled by an exponential function of the quantization value and the function is justified by preliminary studies.

3. After modeling the temporal distortion and quantization distortion, the overall quality of streaming video is modeled by a novel video quality metric - *distorted playable frame rate*, which is represented by a multiplicative function of temporal distortion and quantization distortion.

4. With the ARMOR quality model, an analytical ARMOR optimization algorithm is derived to maximize the quality of streaming video with FEC and media scaling under the capacity constraint.

5. Four variants of ARMOR are studied for different types of FEC and media scaling. Specifically, they are Media Independent FEC with Temporal Scaling (MITS), Media Independent FEC with Quality Scaling (MIQS), Media Independent FEC with Temporal and Quality Scaling (MITQS), and Media Dependent FEC with Quality Scaling (MDQS).

6. Two different Temporal Scaling methods are studied: Pre-Encoding Temporal Scaling (PETS) and POst-encoding Temporal Scaling (POTS), differentiated by the order of encoding and scaling. Their performance and characteristics are studied.

7. The practical length of a Group of Pictures (GOP) is studied to reduce the time that the ARMOR algorithms require to optimize FEC and scaling. The study includes impact of GOP length on video encoding and streaming. Guidelines for GOP length are provided.

8. A user study is conducted to measure video streaming quality with different video content, scaling methods, repair methods and packet loss rates. The high correlation between the video quality estimated by the ARMOR model and user perceptual quality is presented.

9. Simulation experiments are designed with Media Independent FEC with

POst-encoding Temporal Scaling (MIPOTS) and Media Independent FEC with Quality Scaling (MIQS) to show the models and algorithms' accuracy in predicting and optimizing video quality with more realistic network and video conditions.

10. A realistic ARMOR-MIPOTS system is implemented with ARMOR module, MPEG encoder/decoder, FEC encoder/decoder, and TCP-Friendly UDP sender/receiver. The quality of streaming video is measured on the system and the performance of the realistic system is compared with analytical experiments and simulations.

1.4 Road map

The remainder of this thesis is organized as follows: Chapter 2 provides background knowledge to the work in this project; Chapter 3 discusses related research in the area of video streaming over the Internet; Chapter 4 describes the analytical video quality models and optimization algorithms; Chapter 5 discusses the user study; Chapter 6 presents the simulation; Chapter 7 describes the implementation of a realistic ARMOR system; Chapter 8 summarizes the thesis; and Chapter 9 discusses potential future work.

Chapter 2

Background

Topics and terminology that will be used in our discussion of the ARMOR system are reviewed in this chapter. At first, typical video compression standards are discussed in Section 2.1. Then, the idea of Forward Error Correction is explained in Section 2.2. The capacity constraint is introduced in Section 2.3. Media Scaling is presented in Section 2.4 and quality measurement methods are discussed in Section 2.5.

2.1 Video Compression Standard

2.1.1 MPEG

Multimedia objects, especially video, are usually big in size. For example, an typical uncompressed DVD frame sequence, with a resolution of 512x384 pixels, a precision of 24 bits per pixel and frame rate of 25 frames per second, requires a capacity of 142 Mbps for streaming, and needs storage space of 885 Mbytes to store a one-minute clip. This size demands compression to reduce storage, processing and network requirements.

The MPEG (Motion Picture Expert Group) is a popular compression stan-

dard for video [54]. There are two classes of compression techniques in MPEG: *intra-frame* compression and *inter-frame* compression. Intra-frame compression uses the similarity in a single picture when compressing similar to the JPEG standard, and inter-frame compression uses the similarity among a sequence of pictures when compressing. The compression benefits of inter-frame are much larger than the compression benefits of intra-frame.

There are three types of frames defined in the MPEG standard: I frames, P frames and B frames. I (intra-coded) frames are intra-frame encoded independently and focus on encoding similarities within a video frame. P (predictive-coded) frames are inter-frame encoded based on motion differences from preceding I or P frames in the video sequence. B (bi-directionally predictive-coded) frames are inter-frame encoded based on motion differences from preceding and succeeding I or P frames. Typically, the size of I frames are larger than P frames and B frames, and B frames are smaller than I or P frames. There is another type: D frame which has DC coefficients only. But a D frame is seldom used and is excluded from our study of MPEG.

MPEG video typically repeats a pattern of I, P, and B frames, known as a Group of Pictures or GOP, for the duration of a video stream. Figure 2.1 shows a sample GOP, where the second I frame in the figure marks the beginning of the next GOP and the arrows indicate frame dependency relationships. Because of the dependencies of the I, P, and B frames, the loss of one P frame can severely degrade the quality of the other P and B frames, and the loss of one I frame can impact the quality of the entire GOP. This implies that I frames are more important than P frames, and P frames are more important than B frames.

Since B frames cannot be decoded until the subsequent I or P frame has

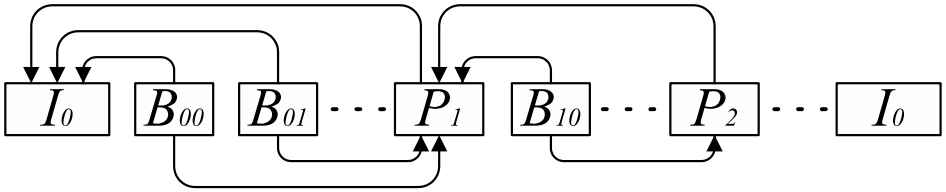


Figure 2.1: A sample MPEG Group Of Pictures (GOP)

arrived,[1] B frames introduce an additional playout delay of one or more inter-frame times. However, this added delay can be controlled by limiting the number of B frames in a row. For example, two B frames in a row, a number typical of many GOPs, in a video encoded at 30 frames per second introduces an additional delay of 66 milliseconds. This work assumes this added delay is tolerable compared to delays induced by the network. However, even in the event that all B frames are discarded, ARMOR presented in this thesis is still valid. An ARMOR extension could use retransmissions in addition to FEC when latency requirements are high. However, the use of retransmission is left as future work and this work focuses on FEC only.

Let N_P represent the number of P frames in a GOP, N_B represent the number of B frames in a GOP, and N_{BP} represent the number of B frames in between an I and a P frame or two P frames[2]. Thus, $N_B = (1+N_P) \times N_{BP}$. Using this notation, a GOP pattern can be uniquely identified by GOP(N_P,N_B). For example, GOP(3,8) indicates the GOP pattern `'IBBPBBPBBPBB'`.

We use the subscripting notation presented in Figure 2.1 to identify individual frames within a GOP. The single I frame of a GOP is referred to as I_0, while P frames are named with P_i, where $1 \leq i \leq N_P$, and B frames are expressed as B_{ij}, where $0 \leq i \leq N_P$ and $0 \leq j < N_{BP}$. For example, P_3 is the

[1] In fact, the following I or P frame is often transmitted before the dependent B frame for this reason.

[2] As in typical MPEG videos, B frames are assumed to be distributed evenly in the intervals between I and P frames.

third P frame, and B_{01} is the second B frame in the first interval of I and P frames.

2.1.2 H.26X

ITU-H.26X are another set of popular compression standards [32]. The coding structure of H.26X is similar to that of the generic codec of MPEG using both intra-compression and inter-compression. Moreover, H.26X has I frames, P frames and B frames as well.

Unlike MPEG, the first version of H.26X, H.261, is aimed at meeting projected customer demand for videophone, video conferencing and other audio-visual services. The bitrate is between approximately 64 Kbps and 1920 Kbps. The coding algorithm of a later version of H.26X, H.263, is similar to that used by H.261, however with some improvements and changes to improve performance and error recovery. Another property of H.26X video is that they typically use much longer GOP patterns than MPEG since they are typically used for video conferencing and have smaller inter-frame variance. The ITU recommendation suggests that a macroblock should be updated at least once every 132 frames [32].

In this project, MPEG is used as the default video standard since it supports a larger range of streaming bitrates and has more available resources such as encoder, decoder, statistical tools and documentation. However, because of the frame types and dependencies, the models and the algorithms, the analysis and the results hold for H.26X as well as other standards.

2.2 Forward Error Correction

Streaming video frames are often larger than a single Internet packet. Because of the dependencies made during compression of the frames, one lost packet can result in the whole frame being undecodable. Moreover, it makes the frames which are dependent on this frame undecodable. For media streaming, the delay and jitter introduced by retransmission could be reduced by buffering. But for the application which has high round-trip time or real-time interaction requirements, retransmission introduces much delay and is not a practical solution for packet loss.

Forward Error Correction (FEC) can be used to recover the frames damaged by packet loss. FEC adds redundant information to the data so that the receiver can re-construct the data from the FEC when there are packet loss. Broadly, there are two types of FEC: media-independent FEC and media-dependent FEC. Media-dependent FEC takes advantages of the data content by adding lower quality data after the original data. When there is a loss, the original data can be replaced by the lower quality data. This method has been widely used in audio streaming [7, 64]. Media-independent FEC does not need to know the content, and instead uses a mathematic algorithm to generate redundant parities from the original data. When there is a loss, the received data and the parities can be used to reconstruct the original data.

2.2.1 Media-Independent FEC (MIFEC)

Reed-Solomon (R-S) coding [72] is a media-independent FEC technique that can be applied at the packet level. As shown in Figure 2.2, an application level video frame is modeled as being transmitted in K packets where K varies with frame type, encoding method, and media content. R-S code adds $(N - K)$

redundant packets to the K original packets and sends the N packets over the network. Although some packets may be lost, e.g. packet 2 in Figure 2.2, the frame still can be completely reconstructed if any K or more packets are successfully received.

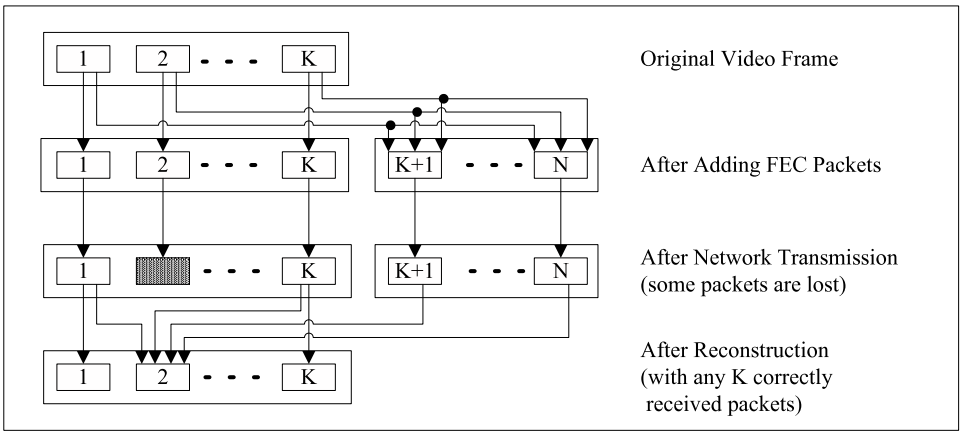


Figure 2.2: Reed-Solomon code

To analyze the effects of FEC on application layer frames, the sending of packets is modeled as a series of independent Bernoulli trials. Thus, the probability $q(N, K, p)$ that a K-packet video frame is successfully transmitted with $N - K$ redundant FEC packets along a network path with packet loss probability p is:

$$q(N, K, p) = \sum_{i=K}^{N} \left[\begin{pmatrix} N \\ i \end{pmatrix} (1-p)^i * p^{N-i} \right] \tag{2.1}$$

Since Equation 2.1 ignores the bursty nature of Internet packet losses, it is proposed to evaluate the impact of this simplifying assumption in this project.

2.2.2 Media-Dependent FEC (MDFEC)

Media-Dependent FEC is applied at the video frame level as in Figure 2.3. Each video frame is encoded with two quality levels: a high quality frame

followed by a low quality frame. Both frames are split into packets and sent into the network. If there is no packet loss in the high quality frame, e.g. the I frame and the first P frame in Figure 2.3 , then the high quality frame will be decoded. If there is any packet loss in the high quality frame while no packet loss in the low quality frame, such as the second P frame in Figure 2.3, the low quality frame will be decoded. Moreover, all the subsequent frames in the GOP, e.g. the last P frame in the figure, which are dependent on this frame, can only be decoded with the low quality level.

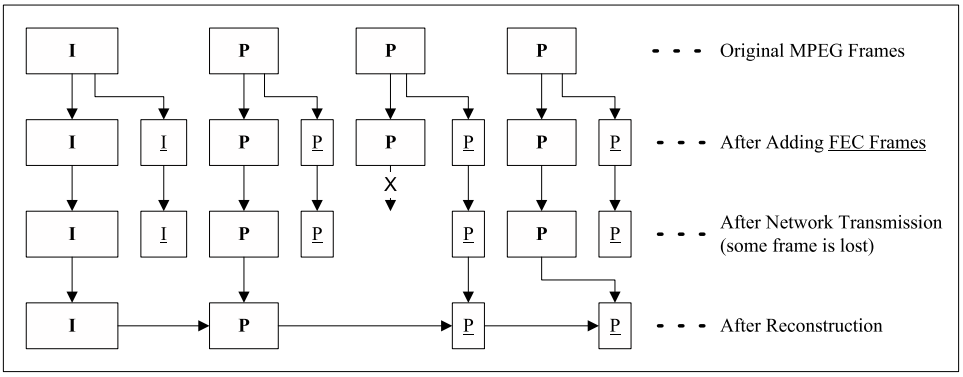


Figure 2.3: Media-Dependent FEC

The effect of MDFEC can be analyzed in the following four cases.

1. When the original frame is received and its reference frame can be decoded with the high quality, it can be played at the receiver side with the high quality.

2. When the original frame is received, but its reference frame can only be decoded with the low quality, it can be decoded with the low quality.

3. When the original frame is lost, but the FEC frame is received, no matter what quality the reference frame is, it can be decoded with the low quality.

4. When the reference frame can not be decoded, any data from the current frame can not be decoded.

Both MIFEC and MDFEC are studied in this thesis, and their performances are compared.

2.3 Capacity Constraint

High quality video typically has a high bitrate when streaming and the capacity of the network serves as a constraint for video streaming. Some potential constraints are TCP-friendly flow limits and ISP capacity limits.

2.3.1 TCP-Friendly Flow

TCP uses the Additive Increase, Multiplicative Decrease (AIMD) method to respond to network congestion. UDP is another popular transmission protocol, but it has no re-transmission mechanism and no congestion control component. By continuing to send data into the network even when the network is saturated, UDP flows can introduce serious congestion. For media streaming, especially real-time interactive applications, which can tolerate some loss but not delay or jitter, both TCP and UDP are not suitable. Other researchers are developing protocols [3, 26, 74] which can adjust their bandwidth to the network conditions smoothly and still remain TCP-Friendly.

A flow is considered to be *TCP-Friendly* if its bandwidth usage in steady-state is no more than a conformant TCP flow running under comparable network conditions (e.g., packet drop rate, round-trip time and packet size). Padhye et al [61] analytically derived the following equation for TCP throughput:

$$T = \frac{s}{t_{RTT}\sqrt{\frac{2p}{3}} + t_{RTO}(3\sqrt{\frac{3p}{8}})p(1+32p^2)} \tag{2.2}$$

where s is the packet size, t_{RTT} is the round-trip time, p is the steady-state packet loss probability, t_{RTO} is the TCP retransmit timeout value. Thus, equation 2.2 provides an upper bound, T, for the TCP-Friendly sending rate. Flows that are not TCP-Friendly can seize a disproportionate share of the network's capacity. Besides being unfair, this type of unresponsive behavior by numerous streaming flows may lead to Internet congestion collapse [11, 25]. Thus, for the Internet to support the future demands for multimedia applications, transport protocols such as [3, 26, 74] that can keep multimedia streaming flows TCP-Friendly can be used in this project.

2.3.2 Maximum Available Bandwidth

Many Internet users connect to the Internet from home. There are several ways for the home user to access the Internet, for example, Dial-up, DSL and cable modem. All these access methods have different bandwidth limits. Dial-up was practically the only choice for home users ten years ago, and it has a bandwidth limited up to 56 Kbps. Cable and DSL modems are the two major choices for higher bandwidth consumers today. Their capacities are several hundred Kbps or a few Mbps. These high speed home connections can support the streaming of high-quality video.

From the view of the end-user, the video-streaming flow does not need to be TCP-Friendly, and it could use all the capacity allocated to the end-user when the streaming is the only application that needs the bandwidth. Take a 1.5M cable-modem user as example. He/she might want to use all the 1.5 Mbps bandwidth in the video-streaming even it is unfair to other flows. This idea has been applied in Microsoft's Windows Media Player as in [58].

Table 2.1 lists some ISP plans in United States as of March 2006. Notice, the table only shows the bandwidth of downlink and uplink bandwidth is

ISP	Type	Bandwidth	Price
AOL	Dial-Up	56 Kbps	$14.95/mon
Verizon	DSL	768 Kbps	$14.95/mon
Verizon	DSL	1.5 Mbps	$21.95/mon
QWest	DSL	1.5 Mbps	$21.95/mon
Charter	Cable	3 Mbps	$19.99/mon
Comcast	Cable	6 Mbps	$42.95/mon

Table 2.1: Some ISP plans

typically much lower.

Moreover, in wireless networks, the loss from the physical layer should not be considered as a signal of congestion for reducing the bandwidth. TCP-Friendly might be inappropriate for this case.

TCP-Friendly constraint is used all through this thesis. However, the models and the algorithms, the analysis and the results hold for ISP capacity limits as well as other capacity constraints.

2.4 Media Scaling

Forward Error Correction can be used to reduce the impact of packet loss, but it adds more data into the network, which already has a limited bandwidth. To preserve the timing aspects of real-time streaming video, the application data rate must be adjusted to the available network bandwidth (i.e., the TCP-Friendly rate or the maximum available bandwidth). Moreover, a multimedia application that is aware of its data dependencies can drop the data that are the least important much more efficiently that a congested router. The method which adjusts the media's data rate is called *Media Scaling*. Some of the Media scaling techniques for video include:

1. Temporal Scaling: The application drops frames before sending them over the network. There are two typical approaches to do Temporal Scal-

ing: *Pre-Encoding Temporal Scaling (PETS)* and *POst-encoding Temporal Scaling (POTS)*. The former drops raw pictures before encoding, and the latter drops the encoded frames.

2. Quality Scaling: The application increases the quantization values and preserves fewer visual details from the original picture.

3. Spatial Scaling: The application reduces the frame size in pixels

This research considers POst-encoding Temporal Scaling, Pre-Encoding Temporal Scaling, Quality Scaling and a combination of Temporal Scaling and Quality Scaling. Since Spatial Scaling is not typically used, it is left as a future work.

2.4.1 Post-Encoding Temporal Scaling (POTS)

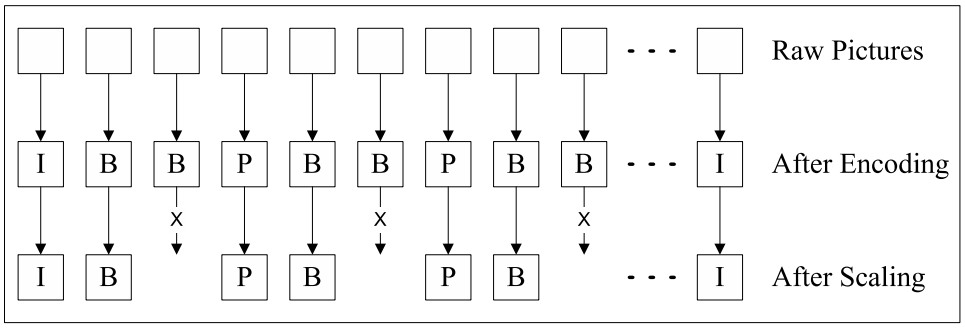


Figure 2.4: Discarding frames in POTS

In Post-Encoding Temporal Scaling, lower priority encoded video frames are discarded prior to the GOP transmission. For instance, with the $GOP(3,8)$ pattern of '`IBBPBBPBBPBB`', the data rate can be approximately halved by discarding all the B frames and only sending '`I--P--P--P--`'.

We use N_{PD} to denote the number of P frames sent in one GOP, and N_{BD} to denote the number of B frames delivered in one GOP ($N_P - N_{PD}$ P frames

are then discarded and $N_B - N_{BD}$ B frames are discarded). For instance, if temporal scaling of GOP(3,8) results in '`I--P--P--P--`' being sent, then N_{PD} is three and N_{BD} is 0. To clarify the temporal scaling decision, we introduce a binary coefficient $D_{\#}$ (e.g. D_I, D_{P_2} or $D_{B_{11}}$) where # can be replaced by I or P or B frame. Specifically, $D_{\#}$ is 0, if temporal scaling discards frame # prior to GOP transmission, and $D_{\#}$ is 1, if frame # will be sent.

While temporal scaling could, in theory, select any of the frames in a GOP to discard, the following set of strategies take into account MPEG frame dependencies and minimize the ill effects of temporal scaling on the quality of the received video:

1. Since B frames depend on I and P frames. B frames are discarded evenly over the GOP before discarding an I or P frame.

2. Since each P frame depends upon the previous P frame or I frame, P frames are discarded from the back (last) to the front of the GOP pattern.

3. Since every frame in a GOP depends upon the I frame directly or indirectly, I frames are never discarded.

Table 2.2 lists all the possible temporal scaling levels for GOP(3,8) with these rules. Each line tells the values of N_{PD} and N_{BD} as well as the scaling patterns and the binary coefficients for that scaling level.

2.4.2 Pre-Encoding Temporal Scaling (PETS)

In Pre-Encoding Temporal Scaling, some of the uncompressed pictures are uniformly discarded before being encoded into video frames. The remaining pictures are then encoded into MPEG frames.

To measure how the raw pictures are discarded, a variable d can be introduced as the the discarding rate, which is the ratio of the number of discarded

POTS Level	N_{PD}	N_{BD}	Scaling Pattern	Binary Coefficient $D_{\#}$
				$I\,B_{00}\,B_{01}\,P_1\,B_{10}\,B_{11}\,P_2\,B_{20}\,B_{21}\,P_3\,B_{30}\,B_{31}$
0	3	8	IBBPBBPBBPBB	1 1 1 1 1 1 1 1 1 1 1 1
1	3	7	IBBPBBPBBPB-	1 1 1 1 1 1 1 1 1 1 1 0
2	3	6	IBBPBBPB-PB-	1 1 1 1 1 1 1 1 0 1 1 0
3	3	5	IBBPB-PB-PB-	1 1 1 1 1 0 1 1 0 1 1 0
4	3	4	IB-PB-PB-PB-	1 1 0 1 1 0 1 1 0 1 1 0
5	3	3	IB-PB-PB-P--	1 1 0 1 1 0 1 1 0 1 0 0
6	3	2	IB-PB-P--P--	1 1 0 1 1 0 1 0 0 1 0 0
7	3	1	IB-P--P--P--	1 1 0 1 0 0 1 0 0 1 0 0
8	3	0	I--P--P--P--	1 0 0 1 0 0 1 0 0 1 0 0
9	2	0	I--P--P-----	1 0 0 1 0 0 1 0 0 0 0 0
10	1	0	I--P--------	1 0 0 1 0 0 0 0 0 0 0 0
11	0	0	I----------	1 0 0 0 0 0 0 0 0 0 0 0

Table 2.2: Temporal Scaling Characteristics

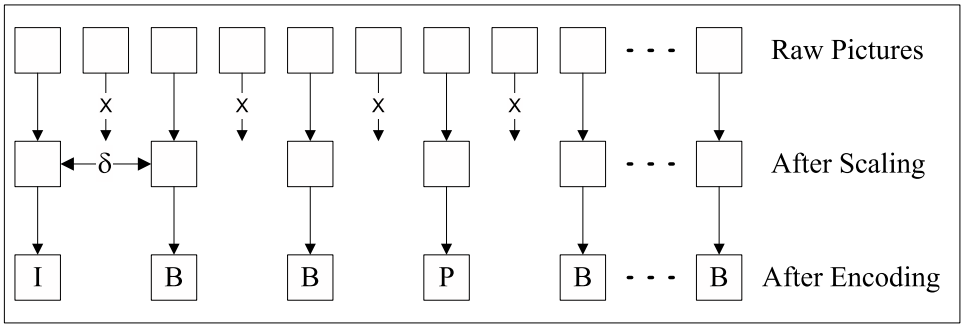


Figure 2.5: Discarding frames in PETS

pictures to the total number of pictures. The value of d varies from 0 to 1, where d=0 means no picture is discarded and all the pictures are sent to the encoder, and d=1 means all pictures are discarded and no picture is sent to the encoder.

Another measurement of the discarding rate is δ, which is the distance between two adjacent pictures sent to the encoder. For example, when d is 0.5, one raw picture is dropped in two pictures, so the δ is 1, since there was one picture between the neighboring encoding pictures. We then have following

relationships for d and δ:

$$d = \frac{\delta}{1+\delta}$$
$$\delta = \frac{d}{1-d} \tag{2.3}$$

To preserve the real-time play rate at the receiver side, PETS needs to slow down its GOP rate. For example, if the discarding rate d is 0.5 (i.e. one picture is discarded every two pictures), the GOP rate should be half of the original rate. Another issue for PETS is that when the raw pictures are discarded before encoding, the similarities among the encoding pictures become less, hence the P frame size and B frame size will increase. Intuitively, the frame size increases as the discarding percentage increases.

2.4.3 PETS vs. POTS

Both PETS and POTS discard frames before transmission and the difference is the order of scaling and encoding. Table 2.3 compares their advantages, disadvantages and other properties.

Pre-Encoding Temporal Scaling	POst-encoding Temporal Scaling
Scaling before Encoding	Scaling after Encoding
GOP pattern does not change	GOP pattern changes by discarding P/B frames
Many scaling levels	Limited scaling levels
GOP duration increases and GOP rate decreases when scaling	GOP duration and GOP rate do not change when scaling
Increasing P, B frame sizes with scaling	Constant frame sizes with scaling
ReCompress before each transmission	Compress once

Table 2.3: PETS vs. POTS

The procedure for scaling with PETS is different than with POTS. PETS discards raw images before encoding them to MPEG frames while POTS encodes all raw images into MPEG frames then discards low priority frames

before transmission.

There are some advantages of PETS. When scaling, PETS does not change its GOP pattern while POTS has to discard P/B frames to change its GOP pattern. In additional, since a GOP usually has limited number of P and B frames, POTS does not have many scaling choices by discarding P/B frames, while PETS has many more scaling choices in choosing different values of the discarding rate d to scale. Besides, when the capacity limit is low, PETS results in smoother video playout since POTS needs to drop P frames from right to left and yields a jerky playout.

On the other hand, PETS also has some disadvantages. When scaling, PETS actually reduces the encoding sampling rate of raw images and increases the actual GOP duration in terms of time. For example, assuming the full motion frame rate is 24 frames per second and the original GOP pattern is `IBBPBB`, then 24 raw images are encoded to 24 MPEG frames in 4 GOPs each second and each GOP takes a quarter second. If PETS is used and the discarding rate d is 0.5 (i.e. every other picture is discarded), then 12 raw images are encoded to 12 MPEG frames each second. Since PETS does not change the GOP pattern, the 12 MPEG frames consists of 2 GOPs and each GOP now takes half second, twice as long as the original. Moreover, to preserve the real-time playout rate at the receiver, PETS needs to reduce its GOP rate since the GOP duration increases. On the contrary, POTS keeps the encoding sampling rate constant and does not need to adjust the GOP rate.

Another potential complication for PETS is that, when the raw pictures are discarded before encoding, the inter-frame coherence decreases, hence the P and B frame sizes will increase. For example, when many raw images are discarded, i.e. δ is larger than a scene, there is no inter-frame compression at

all hence and P and B frames essentially become intra-compressed I frames.

A final consideration for PETS is that it needs to work with raw images instead of MPEG frames as does POTS. Thus, live video, where the raw images can be discarded before compression, is a natural fit for PETS, but pre-encoded video must be decoded before PETS. POTS works equally well with live or pre-encoded video.

Both POTS and PETS are studied in this thesis.

2.4.4 Quality Scaling

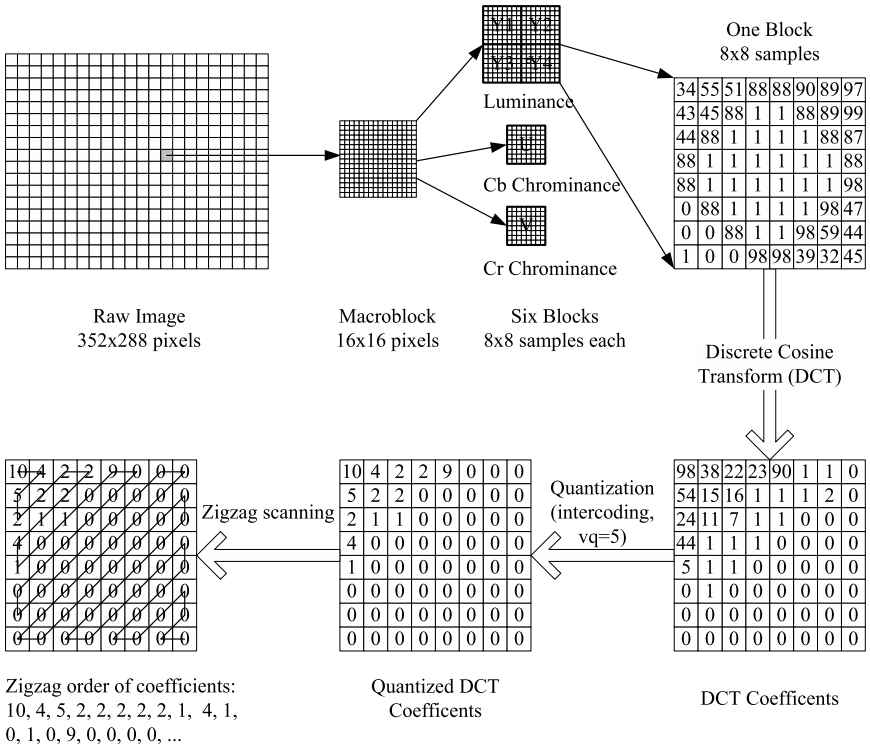


Figure 2.6: MPEG encoding (from [30, 54])

In the MPEG standard, the basic element of each frame is the macroblock [54], consisting of 16x16 pixels. Each macroblock consists of six 8x8 blocks of samples: four luminance blocks and two chrominance blocks. These blocks then are decomposed into a weighted sum of spatial frequencies by the

discrete cosine transform (DCT). A DCT coefficient is quantized by dividing by a positive integer, which is called the *quantization value*, and rounding to the nearest integer. By using high quantization values, MPEG can be encoded in low precision and transmitted with fewer bits. This can be used to reduce the bandwidth for video streaming in a technique called *Quality Scaling*.

Some proposed ways to do quality scaling include:

1. *Adaptive Quantization Value*: Assuming the encoder is aware of the available bandwidth, it can adapt its quantization level according to the network situation. When the bandwidth is limited, the encoder uses a high quantization level making the encoded video small and the quality low. When the bandwidth is not limited, the encoder uses a low quantization level making the video stream large and the quality high.

2. *Signal Noise Ratio Scalability*: It encodes a video clip into multiple layers with different quantization levels. If the bandwidth is limited, only the base layer is sent, and if there is enough bandwidth the higher level layers will be sent too. If the decoder only receives the base-layer information, the decoder displays the lower quality video. If the decoder receives some enhancement layer(s), the decoder gets higher accuracy DCT coefficients by adding the enhanced layers DCT residual to the base-layer, hence displaying a higher quality video.

3. *MPEG-4 Fine Granularity Scalability (FGS)* [44]: The objective of FGS is to optimize the video quality over a network channel, whose capacity varies over a wide range. FGS encodes a video sequence into two layers: base layer and enhancement layer. Different from other discrete layering methods, FGS provides continuous scalability using partial enhancement, where the enhancement bitstream can be truncated into any

number of bits within each frame.

4. *Scalable MPEG (SPEG)*: Similarly to MPEG4-FGS, [41] introduces a simple scalable compression format. Basically, SPEG is a simplified FGS version based on MPEG-1, with every DCT coefficient divided into four layers: one base layer and three advance layers.

   At the receiver/player side, the layers are reversed to reconstruct the original MPEG video where zero is used instead when some advance layer(s) is (are) absent. This is analogous to using a high quantization value during MPEG encoding.

   Since SPEG needs to use extra header information to indicate layer information, there is a 15%-25% overhead [41].

Adaptive Quantization Value and *SPEG* are studied in this thesis. However, the ARMOR quality model and optimization algorithm are independent of the scaling techniques since it only takes the relationships among scaling level, encoding bitrate and video quality.

2.5 Quality Measurement

Because of the compression mechanism of MPEG, MPEG streaming can tolerate some loss, but the loss still impacts the quality. To evaluate the efforts of different video streaming mechanisms and to find the best tradeoff of FEC and media scaling, it is necessary to have measurements of video quality.

2.5.1 Network Goodput

The *network goodput* is the amount of received data which can be used to play the video. It only provides the network level view about how well the video is

streamed. Its advantages are simplicity and application independence. But it does not differentiate streaming video and other network applications such as FTP.

2.5.2 Playable Frame Rate

The *playable frame rate* is how many frames are playable at the receiver side in one second. When the only scaling used is Temporal Scaling, the playable frame rate is a good measure of the video quality. It is simple to calculate in an analytic model accounting for the dependencies among video frames. But when quality scaling is used, the playable frame rate is not an effective measurement of video quality, since two different video clips can have the same frame rate but different picture qualities.

2.5.3 Peak Signal Noise Ratio (PSNR)

PSNR is a popular measurement for measuring video quality when using quality scaling. PSNR compares the difference between the original frame and the decoded frame pixel by pixel. Equation 2.4 gives the equation for PSNR, where $D(x, y)$ is the pixel in the decoded frame and $O(x, y)$ is the original frame.

$$\begin{aligned} PSNR &= 20log(255/MSE) \\ MSE &= \sqrt{\tfrac{1}{N}\sum\left(D(x,y) - O(x,y)\right)^2} \end{aligned} \tag{2.4}$$

However, PSNR does not take into account human vision, and thus is not always a good measurement for the perceived video quality. Besides, since PSNR measures a video by averaging the PSNR of each frame, it is not accurate when there is a frame loss.

2.5.4 Video Quality Metric (VQM)

Better objective measurements use other information such as spatial information, edge energy, temporal information and motion energy as well as PSNR. Video Quality Metric (VQM) is such an objective measurement developed by ITS [65]. It shows a high correlation with subjective video quality assessment and has been adopted by ANSI as an objective video quality standard. The factors that impact quality include blurring, jerky or unnatural motion, global noise and block distortion.

Typically, VQM takes the original video and the processed video as inputs. At first, VQM uses calibration techniques which include spatial alignment and temporal alignment. Then VQM extracts perception-based features in terms of seven parameters, where four are based on features from spatial gradients of the Y luminance component, two are based on features extracted from the vector formed by the two chrominance components and one is based on the product of features that measure contrast and motion.

The following list gives more details about these parameters:

1. *si_loss* detects a decrease or loss of spatial information.

2. *hv_loss* detects a shift of edges from horizontal and vertical orientation to diagonal orientation.

3. *hv_gain* detects a shift of edges from diagonal to horizontal and vertical orientation.

4. *si_gain* detects the improvements to quality that result from edge sharpening or enhancements.

5. *chroma_spread* detects changes in the spread of the distribution of two-dimensional color samples.

6. *chroma_extreme* detects severe localized color impairment.

7. *ct_ati_gain* detects the amount of spatial details and the amount of motion.

Then VQM uses a linear combination of the seven parameters as in the following equation:

$$\begin{aligned} VQM = & -0.2097 * si\_loss - 2.3416 * si\_gain \\ & + 0.5969 * hv\_loss + 0.2483 * hv\_gain \\ & + 0.0192 * chroma\_spread + 0.0076 * chroma\_extreme \\ & + 0.0431 * ct\_ati\_gain \end{aligned} \tag{2.5}$$

By this equation, VQM produces a distortion value between 0 and 1. A value of 0 means the quality of the processed video is as good as the original video and a value of 1 means the processed video has really poor quality compared to the original video.

2.5.5 Subjective Measurement

Ideally, the best quality measurement should be the user perceptual quality since it represents the point of view from the end user. Typically, subjective methods invite groups of people to watch the video and evaluate the video quality. However, subjective measurement can be a very time consuming and thereby expensive work. Moreover, different users can have different opinions on the same streamed video, and even the same user can have different opinions on the same clip under different viewing conditions. These make the subjective measurements more difficult to accurately gather.

ITU-R developed a set of standards [37] to perform subjective assessments and these standards are widely accepted for determining the perceptual video

quality [2, 45, 84]. These standards are briefly described as following:

1. The double-stimulus impairment scale (DSIS) method is designed to evaluate either a new system or the effect of a transmission path impairment. A user session includes a series of videos in random order and with random impairments covering all required combinations. The user is first presented with an unimpaired reference, then with the same video degraded. Following this, the user is asked to vote on the second, comparing it to the first. At the end of the series of sessions, the mean score for each test condition and test picture is calculated. A five-grade degradation scale is recommended: imperceptible; perceptible, but not annoying; slightly annoying; annoying; and very annoying.

2. The double-stimulus continuous quality-scale (DSCQS) method is used for evaluation of a new system or of the effects of transmission paths on quality. In a session, the user is presented with a series of picture pairs (internally random) in random order, and with random impairments covering all required combinations. Each pair of sequences are from the same source, but one via the process under examination, and the other one directly from the source. At the end of all the user sessions, the mean scores for each test condition and test picture are calculated. The users are simply asked to assess the overall picture qualities of both sequences by inserting marks on two continuous vertical scales.

3. The single-stimulus (SS) method presents a single image or sequence of images and the assessor provides an index of the entire presentation. This method can use the same five-grade impairment scale as in the DSIS method.

4. The stimulus-comparison (SC) method displays two images or sequences

of images and the viewer provides an index of the relation between the two presentations. Users can assign the relation between members of a pair to one of a set of seven comparisons: much worse; worse; slightly worse; same; slightly better; better; and much better.

5. The single stimulus continuous quality evaluation (SSCQE) method continuously measures the subjective quality of digitally coded video continuously, with users viewing the material once, without a source reference. It uses an electronic recording slider handset connected to a computer to record continuous quality assessment from the users.

6. The simultaneous double stimulus for continuous evaluation (SDSCE) method presents two sequences at the same time: one is the reference, the other one is the test condition. The two sequences are usually displayed side by side on the same monitor and the users are aware of which is the reference. They are requested to rate the difference by moving the slider of a handset-voting device, while they are viewing the sequences, throughout the total duration.

In this thesis, playable frame rate is used for Temporal Scaling only approach, a new video quality metric, *distorted playable frame rate*, is developed based on both playable frame rate and VQM, and a subjective user study is conducted to show this new metric is highly correlated to user's perceptual quality.

Chapter 3

Related Work

This chapter introduces related research work, covering two research topics corresponding to major parts of our work: *Media Repair*, mainly on the sender side, and *Media scaling*, specifically Temporal Scaling and Quality Scaling.

3.1 Media Repair

Continuous media has different timing requirements and loss tolerance than traditional network applications such as Web browsing and file transfer. Retransmissions are used to replace lost packets for traditional network applications since they typically do not have strict timing requirements and do have strict loss requirements. But retransmission is not always best for streaming media, especially interactive applications since retransmission can add delay and delay jitter. For streaming media, other repair technologies are also used.

3.1.1 Media Repair Taxonomy

Perkins *et al.* [64] surveyed a number of packet-loss repair techniques for streaming audio applications. They summarize the taxonomy as in Figure 3.1.

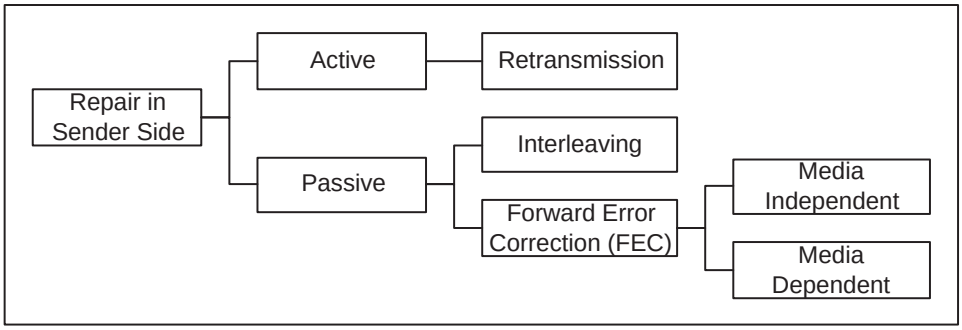


Figure 3.1: Taxonomy of media repair (from [64])

The repair techniques may be classified into active retransmission and passive source-channel joint coding. Passive coding can be further divided to interleaving and forward error correction (FEC), where FEC could be media dependent or media independent. These repair techniques can also be used in video transmission.

3.1.2 Retransmission

The International Telecommunication Union claims that one-way delays of over 300 milliseconds result in poor quality for interactive audio applications [36]. Hence, interactive applications do not typically use retransmission to repair loss. However, for low end-to-end delays, retransmission may be used for loss recovery.

In [21], Feamster *et al.* selectively retransmit only the most important data in the bit-stream assuming the latency requirements do not permit retransmission of all loss data. The server listens for requests on an RTSP port and streams data to the client via RTP. The client gives feedback to the server such as loss, round-trip time and retransmission requests. Then, the server uses a TCP-friendly congestion control mechanism to adjust its bitrate and selectively does the retransmission.

In [17], Dempsey *et al.* defined a Partially Error-Controlled Connection

(PECC) service under which the user submits application- and end-system-specific parameters to coordinate the protocol's use of data retransmissions for error recovery with latency concerns. Basically, if a packet arrives out of sequence, the protocol will decide if there is time to retransmit. If so, the data will be retransmitted, if not, the protocol will discard the packet and other related data.

3.1.3 Interleaving

Interleaving resequences the data units before transmission and re-organizes them at the receiver side to disperses the effect of packet losses. Interleaving is commonly used in memory technology and hard disk technology where the bursty data corruption is common and the data units are small. In media streaming, when the data unit is smaller than the packet size and end-to-end delay is unimportant, interleaving is also useful. Technically, interleaving can not repair packet losses, but rather can reduce the effects of loss by dispersing the loss.

In [92], Zhu *et al.* proposed a video interleaving approach that ameliorates the effects of frame loss by spreading out the bursty effects of loss. The sender first re-sequences data frames before transmission to help distribute loss, and returns the data to their original order at the receiver. They apply the approach to MPEG and evaluate the benefits of interleaving to perceptual quality with user studies. The results show that interleaving can add a small amount of delay and bandwidth overhead, while significantly improving the perceptual quality of Internet video.

In [43], Lee *et al.* propose Interleaved Source Coding (ISC), which is based on an optimum interleaving of predictive video coded frames transmitted over a lossy network. Briefly, ISC divides a raw moving picture sequence into

two sub-sequences which are then encoded separately using the same video encoder. Then, a Stream Merger merges the encoded sub-sequences into a single packet stream in the original-sequence frame order for transmission. The decoder side's Stream Interleaver separates the incoming packets into two sub-streams which are then decoded independent of each other and the Sequence Merger finalizes the process by merging the sub-sequences' frames into the original order for playback. The results show this new method provides clear resilience against packet losses when compared with the traditional (without interleaving) approach.

3.1.4 Media Independent FEC

As discussed in Section 2.2.1, media independent FEC produces additional packets for transmission to repair the losses of original data packets.

In [14], Cho *et al.* developed an adaptive forward error correction protocol that provides a reliable communication service for real-time traffic over satellite networks.

In [50], Mayer-Patel *et al.* use FEC at the packet level to protect the MPEG frames. They have a general analytical model for predicting the reconstructed frame rate of an MPEG stream with FEC. They then use the derived adaptive FEC scheme to study the optimal rate allocation between higher frame rate or high protection with FEC.

In [39], Kang *et al.* studied the performance of FEC-based streaming and provide additional insight into how FEC overhead rate affects the performance of scalable video streaming under dynamically changing network packet loss. Through analytical investigation, they derive the relationship between packet loss, FEC overhead, and utility of received video, and propose a simple control mechanism that adjusts the amount of FEC based on packet loss information.

The results show that the FEC control allows the application to maintain high end-user utility and achieve better quality of video at the receiver.

3.1.5 Media Dependent FEC

Media-dependent FEC uses information in the original content and adds lower quality data after the original data. When the primary frame is lost, the lower quality frame is displayed. Media-dependent FEC is also called as Multiple Description Coding (MDC).

In [46], Liu *et al.* transmit a small, low-quality redundant frame after each full-quality primary frame. In the event the primary frame is lost, they display the low-quality frame rather than display the previous frame or retransmit the primary frame. They simulated the effect of network data loss on MPEG video clips with their media-dependent FEC approach. They also conducted user studies to experimentally measure users' opinions on the quality of the video streams in the presence of data loss, both with and without their redundancy approach. They found media dependent FEC can greatly improve the user perceptual quality of video in the presence of network data loss with an overhead of approximately 10% of the original frame.

The Priority Encoding Transmission (PET) developed at ICSI [1, 12] specifies a different priority for each frame type. According to the assigned priority, PET generates a different amount of redundancy for the segments and disperses user data and redundancy onto several subsequent packets. Typically, I frames are protected with a higher amount of redundancy than P frames, which are protected by a higher amount of redundancy than B frames.

In [8], Bolot *et al.* developed a scheme in which video packet includes redundant information about previous packets. The amount of redundant information is adjusted over time depending on the network conditions. The

redundant information is made up of the macro blocks that are sent in previous packets, but encoded with a lower quality.

In [28], Fumagalli *et al.* apply a novel sequence-based error concealment algorithm to a MDC video coding system that generates a High-Resolution and a Low- Resolution description. In order to recover a loss in the current frames the algorithm takes into account not only the spatial neighboring of the region to which correspond the data loss, but it looks also at what will happen in a significant number of future frames looking at both the High-Resolution and the Low-Resolution descriptions. The sequence-based error concealment algorithm presents good performance when it is applied to video streams.

3.1.6 Combinations

Since each of the previous repair techniques has its own advantages and disadvantages, some of the repair methods can be used together to get combined repair techniques. In [33], Girod *et al.* use retransmission and FEC together for H.263. The results show that for the same PSNR, the net bit rate for video can be significantly reduced when a combination of retransmission and FEC is used. In [68], Qiu *et al.* use interleaving in the physical layer, FEC in the wireless ATM adaptation layer and selective retransmission in the network layer. The results show the approach can randomize bursty errors, achieve optimal error correction and increase bandwidth. In [75], Rhee *et al.* use FEC to protect the most important periodic frames and retransmit lost packets within *periodic temporal dependency distance (PTDD).*

3.1.7 Our approach

FEC, including media independent FEC and media dependent FEC, is studied in this research. Retransmission potentially has a high latency and the retrans-

mitted packet is useless if it arrives after the real-time constraint. Interleaving does not really repair the packet losses and it introduces a significant delay if the data unit size is not small. The delay introduced by FEC is on the order of the interval of the media frame, which is around 30 ms, since FEC is always added over a single frame and sent after each frame. In this research, the media independent FEC is applied at the packet level for each frame. Based on the importance of the frame and the frame size, the number of FEC packets for each type of frame can be determined. Media dependent FEC is also studied in a similar method and compared with media independent FEC in this research.

However, the proposed ARMOR still works for other repair approaches with slight modification. For example, when retransmission is used, the video quality can be estimated as a function of the retransmission policy and scaling level and the streaming bitrate can also be approximated. Then, an exhaustive search can be used to optimize the retransmission and the scaling level to yield the best video quality.

3.2 Media Scaling

As discussed in Section 2.3, to preserve real-time streaming media playout, streaming servers must scale back their streaming data rate to match the TCP-Friendly data rate or the capacity constraint. This proactive data rate reduction by the multimedia server is called *media scaling*. There are several types of media scaling, for example: temporal scaling, quality scaling, Signal Noise Ratio (SNR) scaling, and spatial scaling.

3.2.1 Media Scaling in Research

Quality scaling encodes a video into multiple layers, which have the same frame rate and frame size but different quantization accuracies. Depending on the available network capacity, only certain layers are sent. In [22, 73], the server keeps a hierarchical set of streams as multiple layers. If more capacity becomes available, more layers are sent. If the capacity decreases, fewer layers are sent. In [55], the server detects the available capacity and chooses the appropriate quantization level for encoding.

Temporal scaling encodes a video into multiple levels, which have the same frame size and quantization accuracy but different frame rates [40]. In [16], Conklin *et al.* compare three typical temporal scaling methods: temporal subband coding (TSB) [67], motion-compensated temporal subband coding (MC-TSB) [81], and motion compensated prediction (MCP) [40]. The results show, MCP provides the best performance in terms of quality and bitrate.

Spatial scaling encodes a video into multiple levels, which have the same frame rate and quantization level but different frame sizes. Benzler *et al.* [5] and Naveen et al [56] encode the video into multi-resolution streams and choose the appropriate one for the current network conditions.

Scaling methods can be used in combination. For example, in [19], Domanski *et al.* use both temporal scaling and spatial scaling methods for MPEG video coding. Scaling methods are also compared in previous research. In [51], McCarthy *et al.* compare the effects of quality scaling and temporal scaling for streaming video in small size. Contrary to existing guidelines, they found that users prefer videos that are temporally scaled than video that are quality scaled and the authors attributed this to small screens tested.

3.2.2 Media Scaling in Commercial Software

The two most popular commercial video streaming software are RealNetworks' Realplayer [69] and Microsoft's Media Player [53].

Realplayer [71] uses SureStream technology to create multiple video streams targeted for various network capacities. Using Real Producer, target bitrates can be selected as well as the video stream appropriate for that bitrate. All of the typical means of scalability (spatial, temporal and quality) are utilized in the SureStream technology. When selecting multiple target bitrates for encoding, each bitrate selection represents an independently decodable stream, which can use different resolutions, frame rates and quality levels optimized for that bitrate.

Windows Media Player (WMP) [53] also uses most of the typical scaling methods. For example, video content can be encoded into different window sizes. Small videos are normally used with low bit rate content, while larger resolution videos provide greater visibility and often use higher bitrate. Frame rate is also used as a scaling method. In general, high numbers of video frames displayed per second provide smoother video motion while using higher bitrates. WMP uses multiple codecs: some are optimized for low bitrate and some are optimized for high bitrate. Usually, the server contains several streams for the same content with different quality levels, and hence different bitrates. When streaming, the server will detect the available capacity and choose the appropriate stream.

3.2.3 Our approach

In this research, temporal scaling and quality scaling are studied. But the ARMOR model and algorithm are independent of the scaling methods. Once the relationship of video bitrate, video quality and scaling level are decided,

ARMOR can be used to optimize the repairing and scaling.

3.3 Media Repair with Scaling

In general, media repair techniques add redundant packets for transmission and increase the probability of network congestion. To reduce the streaming bitrate under the capacity constraint, media scaling techniques need to be used with media repair approaches.

In [50], Mayer-Patel *et al.* use a TCP-friendly protocol in the transport layer to decide the data rate and use an analytical model to choose the appropriate FEC amount and GOP rate to yield an optimal playable frame rate.

In [41], Krasic *et al.* use TCP to transmit the data, so the data rate always satisfies the capacity constraint and the packet loss can be repaired by retransmission. The server assigns priorities to the data units and streams them according to their priorities and available network capacity.

In [80], Tan *et al.* create a bandwidth scalable compression scheme that produces individually decodable packets of equal importance. Their approach uses 3D subband decomposition and data partitioning in the subband coefficient domain to provide error protection and progressive quantization to provide bandwidth-scalability.

Our research is inspired by Mayer-Patel *et al.*'s work [50], which captures the temporal relationships between I, P and B frames and uses the model to optimize the amount of FEC packets added to each type of frames. However, ARMOR is different for the following reasons:

1. First, their model makes all the P frames the same when capturing the temporal relationships. This is not true since the P frame in the tail of the GOP is dependent on more frames than the P frame at the head

of the GOP and has a lower playable probability. The ARMOR quality model more accurately captures the dependencies of P frames.

2. Second, the parameters for the FEC packets of their work are in the real number domain, making it impractical. For example, the number of FEC packets for an I frame can not be 2.3 since an integer number of packets must be sent. It is possible to convert the real number to integer, for example 2.3 can be converted to 3, but about a half packet is wasted for each frame on average. Moreover, searching for a solution can be slow in the real number domain. ARMOR uses integer parameters, which are more accurate, and have a smaller search space.

3. Third, and most importantly, their work does not consider any practical scaling methods to reduce the streaming bitrate. They assume the streaming can reduce the GOP automatically with changing the streaming behaviors. ARMOR studies different scaling approaches and adjust the streaming characteristics with the scaling methods.

4. Fourth, their work only studies media independent FEC, but ARMOR studies both media independent FEC and media dependent FEC.

Chapter 4

Analytical Models and Optimization Algorithms

This project is designed to improve the quality of video streaming when there are bitrate constraints. In brief, packet loss is repaired by adding redundancy, and the bitrate is reduced by media scaling.

Different combinations of media scaling and forward error correction are studied for video streaming. Specifically, they are:

1. Media Independent FEC with Temporal Scaling (MITS), which discards frames during media scaling and adds redundant packets for repairing;

2. Media Independent FEC with Quality Scaling (MIQS), which reduces video quality during media scaling and adds redundant packets for repairing;

3. Media Independent FEC with Temporal and Quality Scaling (MITQS), which drops frames and reduces video quality during media scaling and adds redundant packets for repairing;

4. Media Dependent FEC with Quality Scaling (MDQS), which reduces

quality during media scaling and adds lower quality frames for repairing.

In the study of MITS, we found that the GOP length is an important factor in determining the optimization speed. So we also study the impact of GOP length in Section 4.2 and provide some guidelines for practical GOP consideration.

In this project, MPEG is proposed as the video compression and streaming standard, however, the ARMOR models and the algorithms apply to other standards such as H.26X.

4.1 Media-Independent FEC with Temporal Scaling (MITS)

In this section, Media-Independent FEC is used to recover lost packets, and Temporal Scaling is used to adjust to the capacity constraint. Two kinds of Temporal Scaling are used in this section: POst-encoding Temporal Scaling (POTS) (see Section 2.4.1) and Pre-Encoding Temporal Scaling (PETS) (see Section 2.4.2).

When Media-Independent FEC is used, the error correction is independent of the content of the packets and the MPEG frame is either fully repaired or is discarded. So when Temporal Scaling is the only scaling method used, we can assume each playable frame has the same quality since their encoding quantization levels do not change. In this case, the playable frame rate, which represents how many video frames can be played at the receiver, can be used to measure the quality of the streamed video.

4.1.1 System Layers

In ARMOR, the system layers and parameters/variables are shown in Table 4.1.

For a streaming session, we assume the network protocol provides loss rates, round-trip times and packet sizes, while the streaming video application provides details on the MPEG frame characteristics. The model and algorithm developed in the rest of this section allows exploration of the effects that various choices of FEC and Temporal Scaling will have on application performance. In particular, the ARMOR layer can adjust the FEC and Temporal Scaling patterns so as to optimize the video quality, which can then be compared to video with typical FEC patterns and to video without FEC.

Layers	Symbols	Descriptions
MPEG Parameters	R_F	The maximum playable frame rate achieved when there is enough capacity and no loss (typical full-motion video rates have $R_F = 30fps$).
	S_I, S_P, S_B	The size of I, P or B frames respectively, in fixed size packets.
	N_P, N_B	The number of P or B frames in one GOP, respectively.
	N_G	The number of frames in one GOP.
Network Parameters	s	The packet size (in bytes).
	p	The packet loss probability.
	t_{RTT}	The round-trip time (in milliseconds).
	T	The capacity constraint (in packets per sec.), limited by the last mile congestion to an ISP or by a TCP-Friendly rate [61].
ARMOR Variables	S_{IF}, S_{PF}, S_{BF}	The number of FEC packets added to each I, P or B frame, respectively.
	N_{PD}, N_{BD}	The number of P or B frames, respectively, sent per GOP after POTS (see Section 2.4.1).
	δ	The distance of between two neighbor encoding raw images in PETS (see Section 2.4.2).

Table 4.1: System Layers and Parameters/Variables

These are the high-level steps of the process: First, working from MPEG frame sizes and adjustable amounts of FEC per frame type, a series of equations is created to characterize the probability of successful transmission and playout for each MPEG frame type. Then, Temporal Scaling and MPEG frame dependencies are incorporated to derive formulas for transmission rate and playable frame rate. Lastly, considering a capacity constraint, the playable frame rate is optimized by adjusting the Temporal Scaling pattern and amount of FEC per frame.

4.1.2 Successful Frame Transmission Probabilities

Given I, P, and B frame sizes, and the distribution of redundant FEC packets added to each frame type, the Reed-Solomon code equation (Equation 2.1)

provides the probability of successful transmission for each frame type, knowing the amount of redundancy added by Media-Independent FEC:

$$\begin{aligned} q_I &= q(S_I + S_{IF}, S_I, p) \\ q_P &= q(S_P + S_{PF}, S_P, p) \\ q_B &= q(S_B + S_{BF}, S_B, p) \end{aligned} \tag{4.1}$$

4.1.3 Capacity Constraint

For given values of the MPEG parameters and ARMOR variables, the total bitrate needed for the video streaming can be estimated, but the value is limited by T, the capacity constraint from the network layer.

$$G \cdot ((S_I + S_{IF}) + N_{PD} \cdot (S_P + S_{PF}) + N_{BD} \cdot (S_B + S_{BF})) \leq T \tag{4.2}$$

4.1.4 Media-Independent FEC with POst-encoded Temporal Scaling (MIPOTS)

As discussed in Section 2.4.1, POTS adjusts the bitrate by discarding low priority encoded video frames prior to the video transmission. For any $GOP(N_P, N_B)$ the Temporal Scaling pattern can be uniquely identified by (N_{PD}, N_{BD}). For example, the Temporal Scaling pattern $(3, 4)$ will send '`IB-PB-PB-PB-`'.

To keep the playout speed at the receiver side the same as that in the original video, ARMOR expresses the GOP rate (GOPs per second) analytically. Subsequently, the ARMOR model computes the playable frame rate using the frame dependency relationships for each of the I, P, and B frame types. Summing the individual playable frame rates provides the total playable frame rate for the streaming application. Then the ARMOR optimization algorithm

can be used with the model to optimize the playable frame rate by varying the POst-encoding Temporal Scaling pattern and the amount of FEC as a function.

GOP Rate

If the GOP rate is decreased in adapting to the current network capacity, the video will appear to run in "slow motion". Thus, the GOP rate, G, must be kept constant in order to maintain the real-time playout speed at the receiver. Temporal Scaling is then used to maintain a constant GOP rate under a reduced network capacity. Given R_F, which is the target full motion frame rate, the GOP rate (specified in GOPs per second during encoding) is:

$$G = \frac{R_F}{(1 + N_P + N_B)} \tag{4.3}$$

Playable Rate of I Frames

Since I frames are independently encoded, the playable rate the I frames is simply the number of I frames transmitted successfully over the network:

$$R_I = G \cdot q_I \cdot D_I \tag{4.4}$$

It is assumed that D_I is always 1 since the I frame is the most important frame in the GOP and losing the I frame impacts the decodability of all subsequent frames in the GOP. While it is possible to scale even more by discarding I frames as well as other types, but the frame rate will then be extremely low (less than 3 frames per second), so this case is not considered further. Hence:

$$R_I = G \cdot q_I \tag{4.5}$$

Playable Rate of P Frames

The first P frame, P_1, can only be displayed when its preceding I frame and itself are received. Thus P_1's playable frame rate is $R_{P_1} = R_I \cdot q_P \cdot D_{P_1}$. Since each subsequent P_i in the GOP depends upon the success of P_{i-1} and its own successful reception, we have:

$$R_{P_i} = R_I \cdot {q_P}^i \cdot \prod_{k=1}^{i} D_{P_k} \tag{4.6}$$

Using the POst-encoding Temporal Scaling rules in Section 2.4.1, P frames are discarded back to front in the GOP and the P frame playable rate is:

$$R_P = \sum_{i=1}^{N_{PD}} R_{P_i} = G \cdot q_I \cdot \frac{q_P - q_P^{1+N_{PD}}}{1 - q_P} \tag{4.7}$$

Playable Rate of B Frames

All N_{BP} adjacent B frames have the same dependency relationship (they depend upon the previous and subsequent I or P frame) and thus these B frames all have the same playable rate.

When a B frame precedes a P frame, the B frame depends only on that P frame. It is not necessary to consider the I or P frames before this P frame since these dependency relationships have already been accounted for in the successful reception probability of the P frame. Thus:

$$R_{B_{ij}} = R_{P_{i+1}} \cdot q_B \cdot D_{B_{ij}} \;\; when\; 0 \leq i \leq N_P - 1 \tag{4.8}$$

When a B frame precedes an I frame, the B frame depends upon both the preceding P frame and upon the succeeding I frame. For these B frames:

$$R_{B_{ij}} = R_{P_i} \cdot q_B \cdot D_{B_{ij}} \cdot q_I \;\; when\; i = N_P \tag{4.9}$$

Finally, the playable B frame rate for all B frames is:

$$R_B = \sum_{i=0}^{N_{PD}} \sum_{j=0}^{N_{BP}} R_{B_{ij}} \tag{4.10}$$

Total Playable Frame Rate

The total playable frame rate is the sum of the playable frame rates for each frame type:

$$R = R_I + R_P + R_B = R((N_{PD}, N_{BD}), (S_{IF}, S_{PF}, S_{BF})) \tag{4.11}$$

For example, when no frames are discarded due to Temporal Scaling, using the above equations for R_I, R_P and R_B, the total playable frame rate, R, is:

$$\begin{aligned} R &= G \cdot q_I + G \cdot q_I . \frac{q_P - q_P^{N_P+1}}{1-q_P} \\ &\quad + N_{BP} \cdot G \cdot q_I \cdot q_B \cdot (\frac{q_P - q_P^{N_P+1}}{1-q_P} + q_I \cdot q_P^{N_P}) \\ &= G \cdot q_I \cdot (1 + \frac{q_P - q_P^{N_P+1}}{1-q_P} + N_{BP} \cdot q_B \\ &\quad \cdot (\frac{q_P - q_P^{N_P+1}}{1-q_P} + q_I \cdot q_P^{N_P})) \end{aligned} \tag{4.12}$$

Optimal Playable Frame Rate

For given values of p, (N_P, N_B) and (S_I, S_P, S_B), the total playable frame rate R varies with the Temporal Scaling pattern and the amount of FEC as a function R$((N_{PD}, N_{BD})$, $(S_{IF}, S_{PF}, S_{BF}))$. In addition, the streaming bitrate is also limited by the capacity constraint.

So the playable frame rate, R, can be optimized using the operations research equation:

$$
\begin{cases}
Maximize: \\
\quad R = R((N_{PD}, N_{BD}), (S_{IF}, S_{PF}, S_{BF})) \\
Subject\ to: \\
\quad G \cdot ((S_I + S_{IF}) + N_{PD} \cdot (S_P + S_{PF}) \\
\qquad + N_{BD} \cdot (S_B + S_{BF})) \leq T \\
\quad 0 \leq N_{PD} \leq N_P, 0 \leq N_{BD} \leq N_B \\
\quad 0 \leq S_{IF} \leq S_I, 0 \leq S_{PF} \leq S_P, 0 \leq S_{BF} \leq S_B
\end{cases} \tag{4.13}
$$

Unfortunately, finding a closed form solution for the non-linear function R is difficult since there are many saddle points. However, given that the optimization problem is expressed in terms of integer variables over a restricted domain, an exhaustive search of the discrete space is feasible.[1] With fixed input values of network and MPEG layer parameters, (p, RTT, s, T), (N_P, N_B) and (S_I, S_P, S_B), the space of possible values for ARMOR variables, (N_{PD}, N_{BD}) and (S_{IF}, S_{PF}, S_{BF}), (subject to the POTS constraints given in Section 2.4.1) can be exhaustively searched to determine the FEC and Temporal Scaling pattern, which yields the maximum playable frame rate under the capacity constraint.

4.1.5 Media-Independent FEC with Pre-Encoded Temporal Scaling (MIPETS)

As discussed in Section 2.4.2, PETS adjusts the bitrate by discarding some of the raw pictures prior the video encoding. For any sequence of pictures, the Temporal Scaling pattern can be uniquely identified by δ, which is the distance between two neighbor encoding raw images. For example, when δ

[1] In practice, using our model to find the best adjusted FEC and POTS pattern for the GOP of '`IBBPBBPBBPBB`' takes about 30 ms on a P-3 800 MHz.

is 1, one of every two raw images is sent to the encoder. In adapting to the limited capacity, δ needs to be larger than 0. However, when δ increases, the sizes of the encoded P and B frames will also increase because there is less chance of inter-frame compression. However, the size of the I frames will not change since I frames use intra-compression only, which reduces the bitrate by compressing the similarity inside the picture.

As in the ARMOR-MIPOTS, ARMOR-MIPETS uses following steps. To play the original video at its regular speed, the ARMOR model expresses the GOP rate (GOPs per second) analytically. Subsequently, the model computes the playable frame rate using the frame dependency relationships for each of the I, P, and B frame types. Summing the individual playable frame rates provides the total playable frame rate for the streaming application. Then, the ARMOR model can be used to optimize the playable frame rate by varying the Pre-Encoding Temporal Scaling pattern and the amount of FEC as a function.

GOP Rate

When capacity is limited and Pre-Encoding Temporal Scaling is used, part of the raw images will be discarded before encoding, and δ will have a value greater than zero. To keep the playout rate at the receiver the same as the capturing rate of video, the GOP rate must be decreased.

Assume the capturing frame rate is R_F and the GOP length is N_G. When the distance between two adjacent encoding pictures δ is greater than zero, only $R_F/(1+\delta)$ of the raw images will be encoded into GOPs, of length N_G. So the GOP rate, as a function of δ, is:

$$G(\delta) = \frac{R_F/(1+\delta)}{N_G} = \frac{R_F}{N_G \cdot (1+\delta)} \tag{4.14}$$

Functions of P and B Frame Sizes

When the raw pictures are discarded before encoding, the similarities among the encoded pictures decreases and, hence, the sizes of P and B frames increases. At the extreme, when δ is large (say, Δ), the P and B frames effectively become the same as I frames. Assuming the frame sizes increase linearly with increasing δ, one can determine the sizes of P and B frames as functions:

$$\begin{aligned} S_P(\delta) &= S_{P0} + (\delta/\Delta) \cdot (S_I - S_{P0}) \\ S_B(\delta) &= S_{B0} + (\delta/\Delta) \cdot (S_I - S_{B0}) \end{aligned} \tag{4.15}$$

where S_{P0} and S_{B0} are the sizes of the P and B frames, respectively, in the MPEG video without PETS. Experiments [89] show curves up to $\Delta = 9$ fit Equation 4.15 well. Notice that the sizes of the I frames do not change with δ since I frames use intra-image compression only.

Successful Frame Transmission Probabilities

Given I, P, and B frame sizes and the distribution of redundant FEC packets added to each frame type, Equation 4.16 provides the probability of successful transmission for each frame knowing the amount of redundancy added by Media-Independent FEC:

$$\begin{aligned} q_I &= q(S_I + S_{IF}, S_I, p) \\ q_P &= q(S_P + S_{PF}, S_P, p) \\ q_B &= q(S_B + S_{BF}, S_B, p) \end{aligned} \tag{4.16}$$

While this equation looks similar to Equation 4.1 on Page 51, the difference is S_P and S_B change as functions of δ here, but they remain constant in Equation 4.1.

Playable Rate of I Frames

Since I frames are independently encoded, the playable rate of the I frames is simply the number of I frames transmitted successfully over the network. With only one I frame per GOP, the playable I frame rate is simply:

$$R_I = G \cdot q_I \tag{4.17}$$

Playable Rate of P Frames

The first P frame, P_1, can only be displayed when its preceding I frame and itself are successfully transmitted. Notice, in Equation 4.1, which computes the successful frame transmission probability, S_P is a function of δ instead of a constant value as in POTS (Section 4.1.4). This change also applies to B frames, too. Thus, P_1's playable frame rate is $R_{P_1} = R_I \cdot q_P$. Since each subsequent P_i in the GOP depends upon the success of P_{i-1} and its own successful transmission, the playable frame rate of each P frame could be expressed by induction:

$$R_{P_i} = R_I \cdot {q_P}^i \tag{4.18}$$

and the playable P frame rate for all P frames is:

$$R_P = \sum_{i=1}^{N_P} R_{P_i} = G \cdot q_I . \frac{q_P - q_P^{N_P+1}}{1 - q_P} \tag{4.19}$$

Playable Rate of B Frames

All B frames in the same interval between an I or P frame have the same dependency relationship and thus these B frames all have the same playable frame rate.

A B frame that precedes a P frame depends only on that P frame. It is not necessary to consider the I or P frames before this P frame since these dependency effects have already been accounted for in the success probability of this P frame. Thus:

$$R_{B_{ij}} = R_{P_{i+1}} \cdot q_B \;\; when\; 0 \leq i \leq N_P - 1 \tag{4.20}$$

When a B frame precedes an I frame, it depends on both the preceding P frame and the succeeding I frame. For these B frames:

$$R_{B_{ij}} = R_{P_i} \cdot q_B \cdot q_I \;\; when\; i = N_P \tag{4.21}$$

Finally, the playable B frame rate for all B frames is:

$$\begin{aligned} R_B &= N_{BP} \cdot \textstyle\sum_{i=0}^{N_P} R_{B_{i0}} \\ &= N_{BP} \cdot G \cdot q_I \cdot q_B \cdot (\tfrac{q_P - q_P^{N_P+1}}{1-q_P} + q_I \cdot q_P^{N_P}) \end{aligned} \tag{4.22}$$

Total Playable Frame Rate

The total playable frame rate is:

$$R = R_I + R_P + R_B \tag{4.23}$$

Using the above equations for R_I, R_P and R_B, the total playable frame rate is:

$$\begin{aligned} R &= G \cdot q_I + G \cdot q_I . \tfrac{q_P - q_P^{N_P+1}}{1-q_P} + N_{BP} \cdot G \cdot q_I \cdot q_B \\ &\qquad \cdot (\tfrac{q_P - q_P^{N_P+1}}{1-q_P} + q_I \cdot q_P^{N_P}) \\ &= G \cdot q_I \cdot (1 + \tfrac{q_P - q_P^{N_P+1}}{1-q_P} + N_{BP} \cdot q_B \\ &\qquad \cdot (\tfrac{q_P - q_P^{N_P+1}}{1-q_P} + q_I \cdot q_P^{N_P})) \end{aligned} \tag{4.24}$$

Optimal Playable Frame Rate

For given values of network and MPEG parameters, p, (N_P, N_B) and (S_I, S_P^0, S_B^0), the total playable frame rate R varies with the Temporal Scaling pattern and the amount of FEC as a function R(δ, (S_{IF}, S_{PF}, S_{BF})). In addition, the streaming bitrate is also limited by the capacity constraint, T.

Our model can be used to optimize the playable frame rate, R, using the equation:

$$\begin{cases} Maximize: \\ \quad R = R(\delta, (S_{IF}, S_{PF}, S_{BF})) \\ Subject\ to: \\ \quad G(\delta) \cdot ((S_I(\delta) + S_{IF}) + N_P \cdot (S_P(\delta) + S_{PF}) \\ \qquad + N_B \cdot (S_B(\delta) + S_{BF})) \leq T \\ \quad 0 \leq \delta \leq \Delta \\ \quad 0 \leq S_{IF} \leq S_I, 0 \leq S_{PF} \leq S_P, 0 \leq S_{BF} \leq S_B \end{cases} \tag{4.25}$$

Please notice there are a couple of differences between this and ARMOR-MIPOTS in Section 4.1.4. First, the frame sizes here change with δ while the ones in MIPOTS are constant. Second, the GOP rate changes with δ but the rate in MIPOTS does not. Finally, every encoded frame is sent to the network but some of the frames in MIPOTS are discarded before transmission to reduce the bitrate.

Similarly to ARMOR-MIPOTS, with fixed input values for network and MPEG parameters, (p, RTT, s, T), (N_P, N_B), Δ, and (S_I, S_P^0, S_B^0), the space of possible values for ARMOR variables, δ and (S_{IF}, S_{PF}, S_{BF}), can be exhaustively searched to determine the FEC and Pre-Encoding Temporal Scaling pattern to yield the maximum playable frame rate under the capacity constraint.

4.1.6 Summary

This section proposes analytic models for TCP-Friendly MPEG streams that capture the dependencies between MPEG frame types and computes the playable frame rate of temporally scaled MPEG video with Forward Error Correction (FEC) in the presence of packet loss. These models are then used to build optimization algorithms to determine the optimal adjustments of FEC and Temporal Scaling, including POst-encoding Temporal Scaling (POTS) and Pre-Encoding Temporal Scaling (PETS), taking into account both current network conditions and MPEG settings.

Unfortunately, finding a closed form solution for the non-linear Operations Research equations is difficult since there are many saddle points. Given that the optimization problem is expressed in terms of integer variables over a restricted domain, an exhaustive search of the discrete space can be considered. However, since the size of the domain is decided by the number of possible Temporal Scaling levels and the GOP length, the GOP length is an important factor in determining the optimization speed. In the next section, we study the impact of the GOP length and try to limit the practical GOP length to save the search time for optimization.

4.2 Study of GOP Length

4.2.1 Overview

The playable frame rate needs to be optimized by an exhaustive search over a restricted domain. The size of the domain is decided by the number of possible Temporal Scaling patterns and the GOP length and a limited GOP length would save the search time during optimization. Hence the GOP length is an important factor in determining the optimization speed. It is necessary to study the impact of the GOP length on static MPEG files and streaming MPEG.

Currently the choice of GOP is mostly an intuitive process. Some researchers use the default GOP pattern that comes with an MPEG encoder. Other researchers have varied the GOP pattern with little concern for the practical ramifications of the specific GOP pattern on delivery of an MPEG video over a lossy network. In [50], the author searches a large range of GOPs to find the optimal GOP for MPEG streaming, which can result in a large number of P frames in one GOP (e.g., 35 P frames). Such a large GOP is seldom seen in real MPEG encoding [23]. In [20], the authors find the number of B frames between two reference frames should be from 1 to 4 while [91] concludes that the number should be varied from 0 to 2. However, the advantage of these proposed dynamic GOP length mechanisms is not significant. To the best of our knowledge, guidelines on how to *practically* choose a GOP has not been presented in any systematic fashion.

The goal of this section is to investigate practical GOP considerations with respect to performance of MPEG encoded video streams, using a a network model with packet loss and capacity constraints. This research consists of two main components – the study of static MPEG video and analysis of streaming

MPEG video. In the static MPEG analysis, the GOP length and pattern are varied to observe the properties of the resultant MPEG file, noting file size, frame sizes and video quality (measured by Peak Signal-to-Noise Ratio, PSNR). In the streaming MPEG analysis, the GOP is varied to provide insight on the impact of these practical GOP choices on the behavior of streaming MPEG with Forward Error Correction (FEC) [89] and Pre-Encoding Temporal Scaling (PETS) in terms of bitrate and video quality (measured by playable frame rate). The two major recommendations from both components of this study are: 1) the number of B frames between two reference frames should be set to two when the video stream does not have severe delay constraints, and 2) the number of P frames should be 5 or fewer as there is little performance gain in setting the number of P frames in the GOP larger than 5.

4.2.2 Static MPEG Files

Methodology

This section considers the impact of GOP length on static MPEG file properties and suggests guidelines for GOP considerations. The analysis uses the following steps:

1. Study the impact of the number of B frames (denoted as N_{BP}) between two reference (P or I) frames on frame size and frame quality (measured by PSNR). This provides a guideline for choosing N_{BP}.

2. Given the N_{BP} guideline, study the impact of the number of P frames in one GOP (denoted as N_P) on frame sizes and frame quality (measured by PSNR). This provides a guideline for choosing N_P.

Nine video clips are used for the experiments, where each video clip has 300 raw images that play out at 30 fps for 10 seconds. The size of each frame

(a) Container

(b) Hall

(c) News

(d) Foreman

(e) Paris

(f) Silent

(g) Coastguard

(h) Mobile

(i) Vectra

Figure 4.1: Screenshots of Video Clips

Motion	Video	Description
Low	Container(CT)	A working container ship
Low	Hall(HL)	A hallway
Low	News(NW)	Two news reporters
Medium	Foreman(FM)	A talking foreman
Medium	Paris(PR)	Two people talking with high-motion gestures
Medium	Silent(SL)	A person demonstrating sign language
High	Coastguard(CG)	Panning of a moving coastguard cutter
High	Mobile(ML)	Panning of moving toys
High	Vectra(VT)	Panning of a moving car

Table 4.2: Video Clips

is 352x288 pixels (CIF). For each video clip, Table 4.2 provides an approximate motion classification according to our previous study [90], an identifying name with an abbreviation code in parentheses, and a short description of the video content. The abbreviations identify the clips in subsequent graphs. Figure 4.1 shows the screenshots of all the 9 video clips. All the experiments use the Berkeley MPEG encoder and decoder[2]. However, the results should hold for other MPEG encoders since the choice of encoder has little impact on compression relative to the impact on compression due to the choice of quantization level and GOP pattern. The quantization values for I, P and B frames are all 3 to yield a high picture quality in every frame.

Study of N_{BP}

Increasing the number of B frames decreases the correlation between the B frames and the frames they reference [30]. Although the exact tradeoff depends upon the nature of the video scene, for a large class of videos a reasonable spacing of references frames is every 1/10th second. This results in a frame

[2]http://bmrc.berkeley.edu/frame/research/mpeg/

pattern of 'IBBPBBPBB...IBBPBBPBB...' and more generally implies that N_{BP} commonly has a value of no more than two. Mayer-Patel *et al.* [50] used the frame rate of 30 fps and a minimum ratio of reference frames to all frames of 1/3, which also implies N_{BP} is less than three. Feng *et al.* [23] extracted video data from DVDs and also found the most common value of N_{BP} is no more than two.

Experiments were conducted by encoding raw images into MPEG videos with different values of N_{BP} and checking the impact on file size (in Mbytes), frame sizes (in Kbytes) and the quality (measured by PSNR, in decibels).

N_P=1	Frame Size (KB)		PSNR (dB)		File Size
N_{BP}	S_P	S_B	Q_P	Q_B	(MB)
0	11.97	N/A	41.1	N/A	5.18
1	14.22	7.65	41.1	36.7	3.87
2	15.22	8.66	41.1	34.7	3.57
3	16.14	9.46	41.1	33.8	3.53
5	17.36	10.60	41.1	32.7	3.57
11	19.89	12.84	41.1	30.9	3.97

a. N_P=1

N_P=4	Frame Size (KB)		PSNR (dB)		File Size
N_{BP}	S_P	S_B	Q_P	Q_B	(MB)
0	12.05	N/A	41.0	N/A	4.19
1	14.17	7.57	41.0	36.6	3.45
2	15.31	8.60	41.1	34.7	3.33
3	15.93	9.42	41.1	33.9	3.35
5	17.35	10.56	41.1	32.6	3.48
11	19.17	12.81	41.1	30.9	3.93

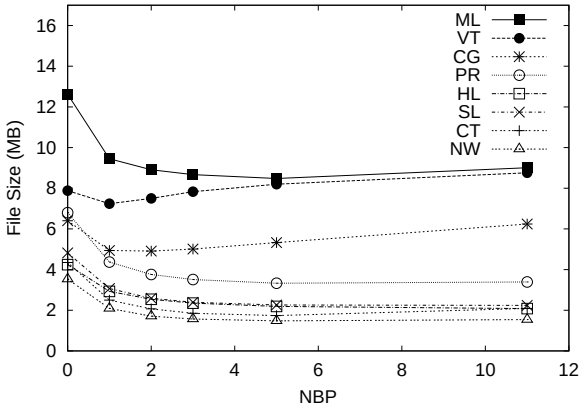
b. N_P=4

Table 4.3: Impact of N_{BP} on MPEG files for *Foreman*

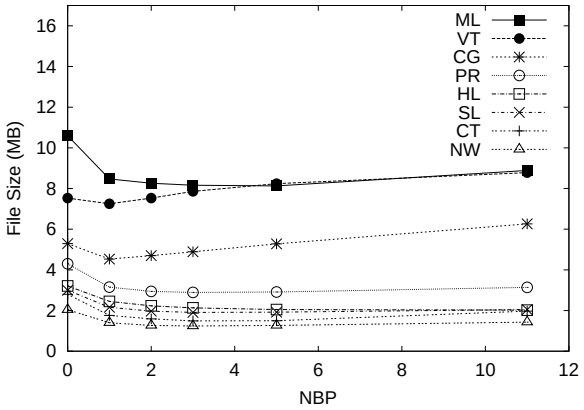
Table 4.3 depicts the frame sizes and PSNR of the *Foreman* video for different N_{BP} sizes with a fixed number of P frames ($N_P = 1$ in Table 4.3.a and $N_P = 4$ in Table 4.3.b). Information on the I frames is not provided since they are intra-compressed only and do not change with GOP pattern. The data in the two tables are similar. This similarity suggests that the impact of

N_P is small (the next Section, Section 4.2.2, explores N_P in more detail). As N_{BP} increases, the quality of the B frames decreases quickly. For example, in Table 4.3.a, the PSNR of the B frames drops dramatically from 36.7 dB to 30.9 dB. Notice that when N_{BP} increases, the sizes of the P and B frames also increase. In both tables, the sizes of the B frames nearly double as N_{BP} goes from 1 to 11 and this also causes the MPEG file size to grow when N_{BP} is above 2. In theory, having more B frames can reduce the MPEG file size since B frames are usually smaller than I frames. However, since the average size of a B frame increases when there are more B frames, the MPEG file does not necessarily have a higher compression rate for a larger number of B frames. In fact, note that the size of the MPEG file is close to the lowest when $N_{BP} = 2$. These facts suggest that although B frames have the highest compression ratio, a large number of B frames in a GOP introduces low inter-frame compression and lower quality. Thus, a guideline is to have N_{BP} close to or equal to two.

Similar experiments were conducted with the other eight videos in Table 4.2. Figure 4.2 shows the impact of N_{BP} on encoded MPEG file size ($N_P = 1$ in Figure 4.2.a and $N_P = 4$ in Figure 4.2.b). In the figures, the x-axes are N_{BP} and the y-axes are the encoded file size in Mbytes. The figures show $N_{BP} = 2$ provides a small file, very close to the minimum size, for all videos. This result does support previous research [20, 91] which discuss that content-based dynamic GOP length can increase MPEG performance. However, the graphs imply the performance improvement is not significant when more B frames are added to the GOP. The PSNR data is not presented for these videos because the results in all cases are very similar to those in Table 4.3 in that the PSNR of the B frames drops dramatically by around 5dB for N_{BP} of three or larger. These results clearly suggest a practical GOP guideline of keeping N_{BP} close to two.



a. $N_P = 1$



b. $N_P = 4$

Figure 4.2: Impact of N_{BP} on MPEG files for the other videos

Another practical constraint for N_{BP} is that for streaming MPEG, B frames can not be decoded until after the arrival of the subsequent I or P frame. This implies latency increases linearly with the number of B frames. For interactive applications, such as a videoconference, the added latency contributes to the end-to-end delay. For typical full-motion streaming (30 fps frame rate), each B frame contributes about 33 ms of delay. In studies of streaming video on the Internet [15] and network delays in general [38], the median round-trip times for a variety of network configurations are around 100 ms. Thus, compared to the round-trip time, one or possibly two B frames may not represent a significant increase the end-to-end delay, while the use of three B frames could

double the end-to-end delay. Thus, a GOP guideline for streaming MPEG is to have N_{BP} as high as the latency tolerates, but no more than 2.

In summary, the number of B frames between two reference frames should be less then or equal to two. This guideline is used in informing all subsequent experiments.

Study of N_P

Similar to section 4.2.2, experiments were run by encoding the raw *Foreman* images into MPEG videos with different N_P values and analyzing the impact on file size, frame sizes and PSNR.

N_{BP}=2	Frame Size(KB)		PSNR (dB)		File Size
N_P	S_P	S_B	Q_P	Q_B	(MB)
0	N/A	8.83	N/A	34.8	4.02
1	15.22	8.66	41.1	34.7	3.57
5	15.31	8.60	41.1	34.7	3.30
9	15.30	8.59	41.0	34.7	3.25
14	15.17	8.60	41.0	34.7	3.23
29	15.22	8.60	41.0	34.7	3.20

Table 4.4: Impact of N_P on MPEG files for *Foreman*

Table 4.4 presents frame sizes of the *Foreman* video clip for different values of N_P ($N_{BP} = 2$). These results show that as N_P increases, the sizes of the P and B frames do not significantly change, nor does the frame quality. Since increasing the GOP length does not impact the frame size and typical P frames are smaller than their referenced I frames, more P frames can reduce the MPEG file size, as shown in the last column of Table 4.4. However, the reduction in file size is not significant.

Similar experiments were conducted with the other eight videos in Table 4.2. Figure 4.3 presents the impact of N_{BP} on encoded MPEG file size ($N_{BP} = 2$). In the figure, the x-axis is N_P and the y-axis is the encoded file

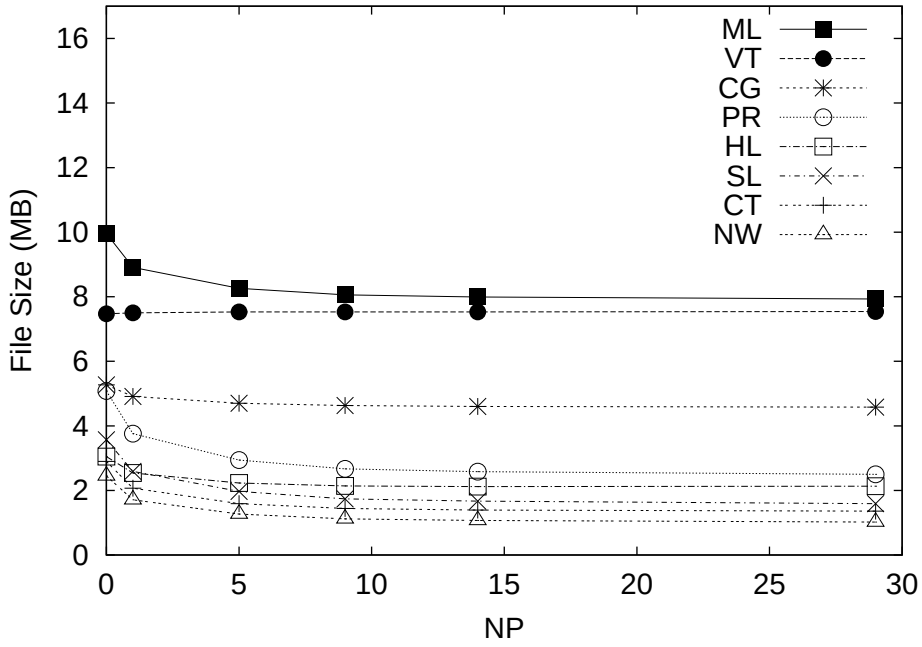


Figure 4.3: Impact of N_P on MPEG files for the other videos

size in Mbytes. More P frames can reduce the MPEG file size, but the reduction is not significant after $N_P = 5$. The corresponding PSNR data is not presented, but the results are very similar to Table 4.4, with the frame quality changing little with increases in N_P.

Another practical constraint associated with the number of P frames is the need to support VCR-like functions (pause, rewind, fast-forward, etc.). To avoid decoding of every frame, response to these functions require access to the I frames, this suggests the GOP length should not be long. For example, if a user wants to pause a movie with a precision of 3 seconds, the GOP length should be no more than 90, and therefore the number of P frames should be at most 90, and more likely at most 30 if N_{BP} is 2.

As a summary, while there are no specific constraints concerning the number of P frames, as a guideline, the number of P frames should be no more than 30. Moreover, while having more P frames can improve the compression

ratio, the benefit is not significant compared to the compression ratio obtained with five P frames per GOP. This guideline is used in informing all subsequent experiments and analysis.

4.2.3 Streaming MPEG

Methodology

This section studies the impact of the GOP pattern on MPEG streaming under conditions of packet loss and limited capacity. Using the guidelines obtained in the static MPEG analysis, the streaming analysis uses the following steps:

1. Use the ARMOR-MIPETS model in Section 4.1.5 to estimate the video quality (measured by playable frame rate).

2. Use the model in the ARMOR-MIPETS optimization algorithm to optimize the video quality.

3. Use the model and algorithms in conjunction with a model of network packet loss and capacity limit to study the impact of GOP length on streaming performance.

ARMOR-MIPETS is presented in Section 4.1.5 in detail. Briefly, with fixed input values for network and MPEG parameters, (p, RTT, s), (N_P, N_B), Δ, and (S_I, S_P^0, S_B^0), the space of possible values for ARMOR variables, δ and (S_{IF}, S_{PF}, S_{BF}), can be exhaustively searched to determine the FEC and Pre-Encoding Temporal Scaling pattern to yield the maximum playable frame rate under the capacity constraint.

Then, the GOP pattern is varied with different values of N_P and N_{BP} used to encode the MPEG stream. For each stream, the frame sizes are extracted and fed into our MIPETS model and algorithm (Equation 4.25 at Page 60) to

find the optimal playable frame rate. By comparing the playable frame rates of different streams, the impact of the GOP pattern on streaming MPEG is analyzed.

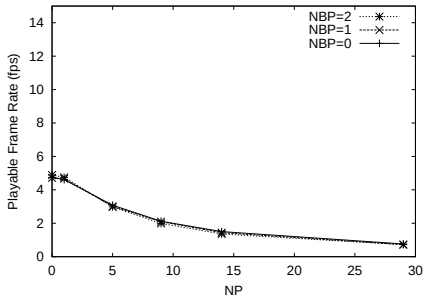
4.2.4 Analysis

Three different FEC choices are considered:

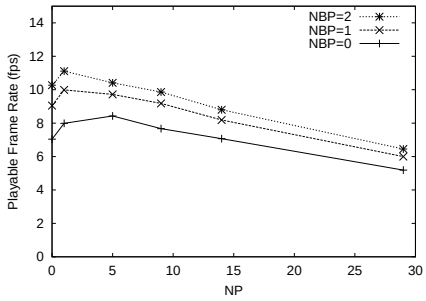
- Non-FEC: The sender adds no FEC to the video.
- 5% Fixed FEC: The sender protects each frame with FEC the size of 5% of the original frame size.
- Adjusted FEC: Before transmitting, the sender uses our model and optimization algorithm (Equation 4.25 on Page 60) to determine the FEC pattern and Temporal Scaling level that produce the maximum playable frame rate and uses these for the entire video transmission.

In all cases, the bitrates used by the MPEG video with added FEC are scaled by PETS to meet the capacity limits. Figure 4.4 shows performance results for a set of experiments with a 1.5 Mbps capacity constraint and with 2% induced modeled packet loss for the video *Foreman*. In the figure, the x-axis is N_P and the y-axis is the playable frame rate. Figure 4.4a shows the playable frame rates for different GOP patterns without FEC, Figure 4.4b is with fixed FEC, and Figure 4.4c uses adjusted FEC. The figures show, fixed FEC is more effective than non-FEC when there is considerable loss since it repairs the loss, preventing degradation in the video quality. In all cases, the mechanism for adjusting FEC searches the space of choices for the best value of FEC and thus yields the best quality.

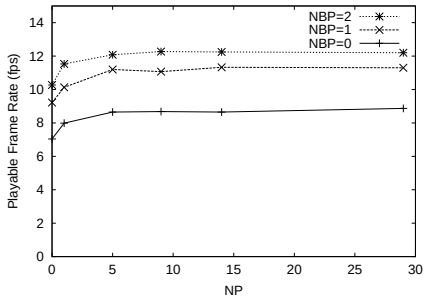
More importantly for the focus of this section, the impact of GOP on streaming MPEG, these figures show results similar to those in the static



a. Non-FEC



b. 5% Fixed FEC



c. Adjusted FEC

Figure 4.4: Streaming *Foreman* with FEC and PETS. Network model has 2% loss and 1.5 Mbps capacity constraint

MPEG study (Section 4.2.2). All three graphs demonstrate that larger values of N_{BP} yield better quality (although delay constraints for interactive applications still limit N_{BP} to be no larger than 2) and there is little to be gained by having N_P greater than 5.

Figures 4.5 depicts the impact of N_P ($N_{BP} = 2$) on streaming MPEG with adjusted FEC for the other 8 videos, where the network model has a 1.5 Mbps capacity constraint and a 2% packet loss is modeled. In the figure, the x-axis

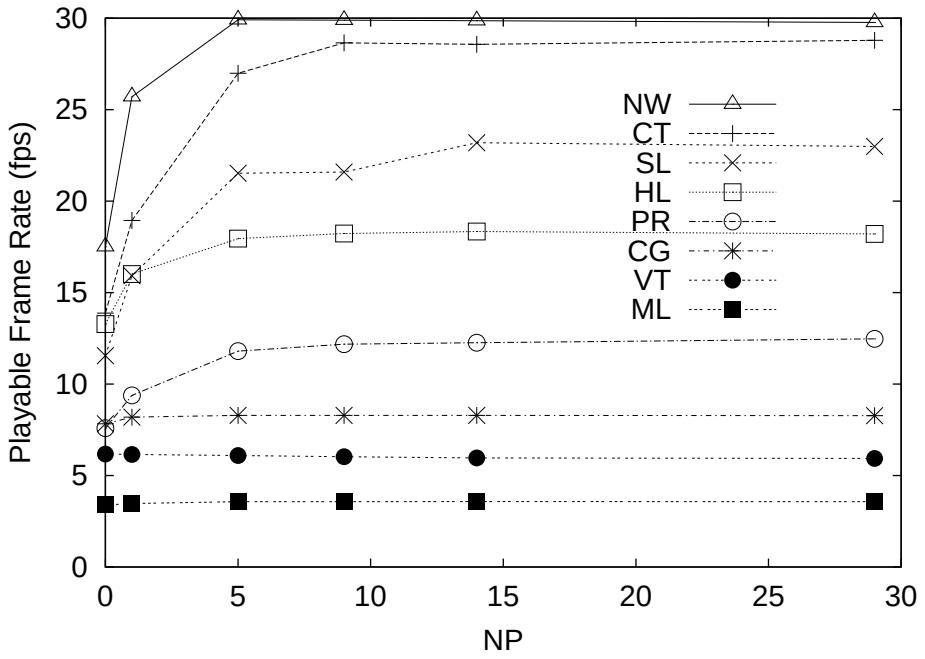


Figure 4.5: Streaming the other 8 videos with adjusted FEC and PETS, 2% loss and 1.5 Mbps capacity constraint ($N_{BP} = 2$).

is N_P and the y-axis is the playable frame rate. The figure shows, for each video, the playable frame rate increases fast from $N_P = 0$ to $N_P = 5$, but does not change much after $N_P = 5$ These results suggest there is little to be gained by having $N_P > 5$.

Figures 4.6 depicts the corresponding δ values in Figure 4.5. In the figure, the x-axis is N_P and the y-axis is δ, the scaling level. It shows, for most cases, δ is less than 2 and our guidelines for GOP length are reasonable. However, for some high-motion and complex scene videos, δ can be large in order to satisfy the capacity constraint. As we discussed in Section 2.4.3, PETS reduces the GOP rate when scaling and it means the effective GOP length (in term of number of raw images) increases. If an I frame is unable to be decoded because of packet loss, there will be a long silent gap in the playout at the receiver side. To solve this problem, a dynamic GOP could be used. Dynamic

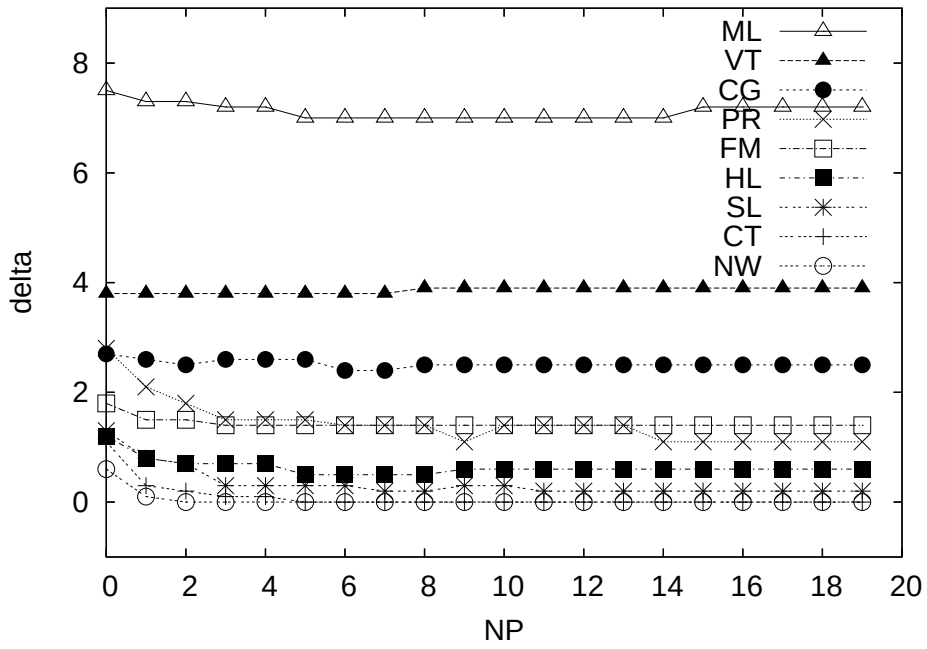


Figure 4.6: Corresponding δ values in Figure 4.5

GOP adjustment is left as future work. From now on, we focus our study on POst-encoding Temporal Scaling (POTS) for Temporal Scaling.

4.2.5 Summary

This section presents an organized methodology to better understand the practical impact of both the GOP length and the detailed GOP pattern on static and streaming MPEG. Utilizing results from experiments and analytic modeling, practical guidelines are put forth for setting the GOP length and selecting an appropriate GOP pattern over a range of MPEG conditions.

In the first set of experiments, raw video images were encoded to MPEG files. These results suggest two guidelines: 1) The number of B frames between two reference frames should not exceed 2; and 2) while there were no specific limitations to the number of P frames in a GOP pattern, there should be no more than 30 P frames in the GOP pattern to support VCR-like functions.

The second phase of our investigation considers the GOP impact when MPEG was sent over a lossy network with Forward Error Correction, which protects packet loss, and Pre-Encoding Temporal Scaling, which satisfies capacity constraints. The optimal MPEG quality occurs when $N_{BP} = 2$ and $N_P \leq 5$. The results suggest two guidelines: 1) The number of B frames between two reference frames should be kept at 2 except when constrained lower by delay constraints; and 2) the number of P frames need not be more than 5.

4.3 Analytical Experiments of MITS

After studying the impact of the GOP length, the playable frame rate can be optimized by a selective search over a restricted domain where a limited GOP length saves the search time significantly.

4.3.1 Analytical Experiments of MIPOTS

In this section, we consider the design of a set of experiments that use ARMOR-MIPOTS analytically to explore the performance of post temporally scaled MPEG video without FEC, with fixed FEC, and with adjusted FEC, where the videos' bitrates are constrained by TCP-friendly data rates.

Overview

Using the formulas in Section 4.1.4, we built a function, `frameRate()` to use Equation 4.11 to compute the playable frame rate with given network characteristics (p, t_{RTT}, s), MPEG properties (N_P, N_B), (S_I, S_P, S_B), Temporal Scaling pattern (N_{PD}, N_{BD}) and amounts of FEC (S_{IF}, S_{PF}, S_{BF}).

Another program was built such that given values of (p, t_{RTT}, s), (N_P, N_B) and (S_I, S_P, S_B) the program iterates through all combinations of Media-Independent FEC (S_{IF}, S_{PF}, S_{BF}) and Temporal Scaling patterns (N_{PD}, N_{BD}). Initially, each combination of FEC and scaling are tested to determine if this combination satisfies the TCP-Friendly rate constraint (Equation 4.2). If this combination does not satisfy the constraint, the search program goes to the next iteration. If the constraint is satisfied, the `frameRate()` function is used to determine the playable frame rate for this FEC and scaling combination. After iterating through all the combinations of FEC and scaling pattern within the constrained search space, the program produces the

maximum playable frame rate, the adjusted FEC (S_{IF}, S_{PF}, S_{BF}), and the Temporal Scaling (N_{PD}, N_{BD}) required to achieve this maximum rate.

Using these programs, the optimal playable frame rates over a range of network and application settings are explored. For each set of network and application parameters, the playable frame rates are compared for MPEG streaming without FEC, MPEG streaming with two different amounts of fixed FEC, and MPEG streaming with adjusted FEC. The following list gives the details about these four FEC choices.

1. Fixed FEC (1/0/0): Each I frame receives 1 FEC packet. This simple FEC pattern protects the most important frame, the I frame. Repairing the I frame is a scheme used by other researchers [21, 75].

2. Fixed FEC (4/2/1): The sender protects each I frame with 4 FEC packets, each P frame with 2 FEC packets and each B frame with 1 FEC packet. This FEC pattern provides strong protection to each frame and roughly represents the relative importance of the I, P and B frames. For the MPEG application settings in Table 4.5, this adds approximately 15% overhead for each type of frame, which is typical for many fixed FEC approaches [34, 35, 47].

3. Adjusted FEC: Before transmitting, the sender uses the programs described previously to determine the FEC and Temporal Scaling patterns that produce the maximum playable frame rate and uses these for the entire video transmission.

4. Non-FEC: The sender adds no FEC to the video.

In all cases, the total bandwidth used by the MPEG video and FEC is scaled to meet TCP-friendly constraints using POTS (Section 2.4.1).

While there are numerous other fixed FEC and MPEG video choices that could be selected, here we only present the analysis of the four representative systems given above. However, the fact that these choices include commonly used FEC patterns and the settings were chosen to capture typical MPEG characteristics justifies this method of performance comparison. Moreover, while other fixed FEC patterns may do as well as adjusted FEC for some MPEG videos under a given set of network conditions, fixed FEC schemes cannot operate effectively over the full range of typical MPEG and network parameters. However, additional comparisons that include other fixed FEC schemes can be found in [86].

System Settings

Table 4.5 presents the system parameter settings for the network and MPEG layers. The MPEG frame sizes were chosen using the mean I, P, B frame sizes measured in [42], and then rounding up the frame size to the nearest integer number of packets. Specifically, the I frame has 25 packets, the P frame has 8 packets and the B frame has 3 packets. A commonly used MPEG GOP pattern, '`IBBPBBPBBPBB`', (GOP(3,8)) and a typical full motion frame rate R_F of 30 frames per second (fps) were used. These settings yield a packet rate of 146 packets per second and a data rate of 1.168 Mbps for the MPEG video. The packet size s, round-trip time t_{RTT} and packet loss probability p were chosen based on the characteristics of many network connections [15, 38, 63]. For all experiments, the parameters are fixed, except for the packet loss probability p, which ranges from 0.01 to 0.04 in steps of 0.001.

Network Layer		MPEG Layer			
t_{RTT}	50 ms	S_I	24.6 Kbytes (25 pkts)	N_P	3 frames per GOP
s	1 Kbyte	S_P	7.25 Kbytes (8 pkts)	N_B	8 frames per GOP
p	0.01 to 0.04	S_B	2.45 Kbytes (3 pkts)	R_F	30 frames per sec

Table 4.5: System Parameter Settings

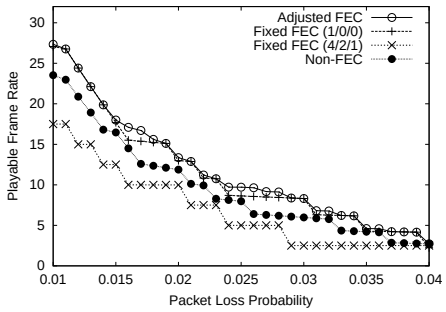
Analysis

Figure 4.7 depicts the playable frame rates for each of the four schemes. For all figures, the x-axes are the packet loss probabilities, and the y-axes are the playable frame rates. For frame rate targets [70]: 24-30 frames per second is full-motion video, 15 frames per second can approximate full motion video for some video content, 7 frames per second appears choppy, and at 3 frames per second or below the video becomes a series of still pictures.

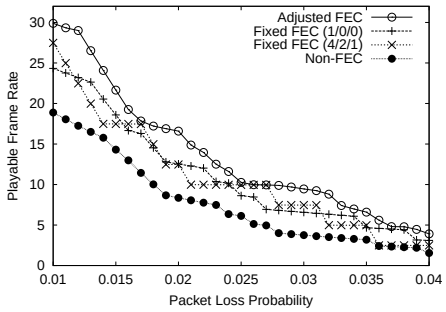
In Figure 4.7, adjusted FEC provides the highest playable frame rate under all network and video conditions. For the typical video size in Figure 4.7b, the benefits of adjusted FEC over non-FEC are substantial, almost doubling the frame rate at 1% loss, and still surpassing the minimum 2 frames per second at 4% loss. The two fixed FEC techniques usually improve playable frame rates over non-FEC video, and FEC(4/2/1) even matches the playable frame rate provided by adjusted FEC for a few loss rates, such as 2.5%.

For smaller video frame sizes in Figure 4.7a, halving the frame sizes in Table 4.5 and doubling the round-trip time to provide an equivalent available bandwidth allows a visual comparison between graphs. FEC(1/0/0) does substantially better, coming closer to the maximum frame rate achieved by adjusted FEC. FEC(4/2/1) does worse with playable frames below the non-FEC scheme. This situation happens because the fixed number of FEC packets added is a larger fraction of overhead for the smaller video frames.

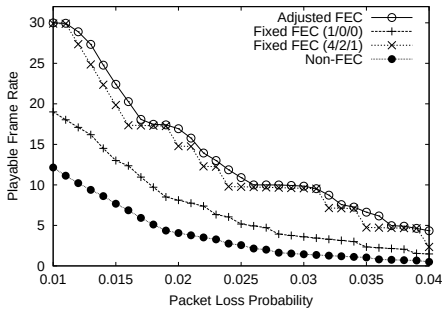
For the larger video frame sizes in Figure 4.7c, created by doubling the



a. Small Frame Size (1/2 those in Table 4.5)



b. Medium Frame Size (as in Table 4.5)



c. Large Frame Size (2x those in Table 4.5)

Figure 4.7: Comparison of Playable Frame Rates

frame sizes in Table 4.5 and halving the round-trip time, FEC(4/2/1) does substantially better and provides close to the maximum frame rate achieved by adjusted FEC. FEC(1/0/0) does significantly worse since it does not provide enough protection for the larger frame sizes. With playable frame rates well below that of adjusted FEC, FEC(1/0/0) still outperforms the non-FEC scheme.

These figures show fixed FEC only works well for specific network and MPEG conditions. For example, FEC(1/0/0) works nearly as well as the

adjusted FEC in Figure 4.7a while FEC(4/2/1) works nearly as well as the adjusted FEC in Figure 4.7c. However, when the network and MPEG conditions change, both fixed FEC patterns chosen are less effective than the more robust adjusted FEC scheme. This general behavior holds for other fixed FEC choices, regardless of the specific input patterns used.

Adjusting FEC

To better explain the benefits of adjusted FEC presented in the previous section, we now analyze how FEC is adjusted for various fixed loss rates.

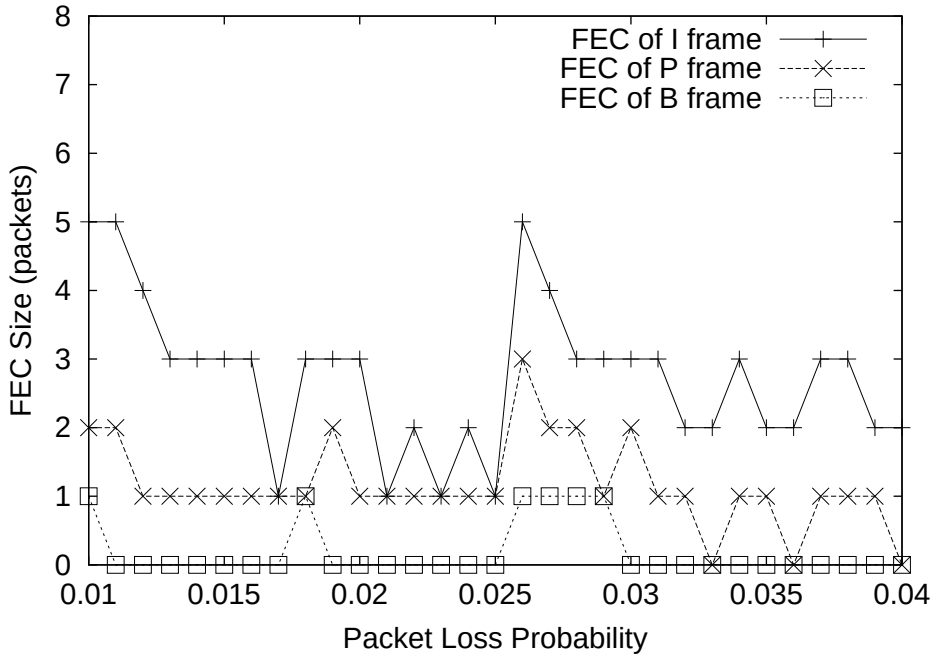


Figure 4.8: Adjusted FEC Pattern

Figure 4.8 gives the breakdown of the adjusted FEC for each I, P, and B frame that produces the maximum playable frame rate versus the loss probability. The fixed FEC approaches are not shown, but they would be represented by horizontal lines since they introduce the same amount of FEC regardless of loss probabilities. For example, FEC(4/2/1) would have a hor-

izontal line at 4 for the I frames, at 2 for the P frames and at 1 for the B frames. In general, without FEC, I frames have a decreasing probability of successful transmission. With adjusted FEC, the most important I frames have the highest transmission probability followed next by the P frames and lastly by the least important B frames. However, there are cases where the best use of FEC is somewhat non-intuitive. For instance, at 1.7% loss, the adjusted FEC scheme reduces the FEC for the I-frames and then increases it at 1.9% loss. This seeming contradiction is because the use of FEC is coupled with Temporal Scaling. In particular, at 1.7% loss, the playable frame rate is higher if four B frames are transmitted (transmitting '`IB-PB-PB-PB-`'), leaving less leftover capacity for FEC. At the increased loss rate of 1.9%, the reduced available bandwidth and higher loss rates makes discarding two more B frames (transmitting '`IB-PB-P--P--`') and using the remaining bandwidth for FEC the right choice for a higher playable frame rate.

Temporal Scaling Pattern

Table 4.6 shows the chosen Temporal Scaling pattern for adjusted FEC as loss probability varies. The '`-`' symbol denotes frames that are discarded by the sender before being transmitted. A B frame is automatically discarded if the following P frame it references is discarded. Although there may be available capacity for the transmission, this B frame still cannot be displayed by the receiver and thus it is discarded. As p increases, the available bitrate under the TCP-Friendly constraint decreases, and the sender discards the less important frames before sending them. The I frames are always transmitted, the P frames are kept as long as possible, and the B frames are discarded before the P frames they reference. In general, an MPEG video with adjusted FEC must discard slightly more frames than the same MPEG video without

FEC. However, the additional packet space saved by the discards can be very effectively used for FEC packets. Temporal Scaling patterns over a larger range of packet loss probability can be found in [86].

p	Adjusted FEC	Non- FEC
0.010	IBBPBBPBBPBB	IBBPBBPBBPBB
0.015	IBBPB-PB-PB-	IBBPBBPBBPB-
0.020	IB-P--P--P--	IB-PB-PB-P--
0.025	I--P--P-----	I--P--P--P--
0.030	I--P--P-----	I--P--P-----
0.035	I--P--------	I--P--------
0.040	I--P--------	I--P--------

Table 4.6: Temporal Scaling Patterns

Note, the Temporal Scaling patterns in Table 4.6 may result in a variable playable frame rate when measured over one GOP, which may impact perceptual quality. Our future work is to incorporate the impact of variance in frame rates, in additional to average playable frame rate, into ARMOR and get the optimal scaling pattern for the best perceived quality. If a low variance is more important than a high playable frame rate, only scaling patterns that evenly distribute the frame discards can be considered.

4.3.2 Analytical Experiments of MIPETS

Similar to the analytical experiments of MIPOTS (Section 4.3.1), we consider the design of a set of experiments that use the MIPETS analytically to explore the performance of pre-encoding temporally scaled MPEG video without FEC, with fixed FEC, and with adjusted FEC, where the videos' bitrates are constrained by TCP-Friendly data rates.

Overview

Similar to MIPOTS, we built a function to compute the playable frame rate with given network characteristics (p, t_{RTT}, s), MPEG properties (N_P, N_B), Δ, and (S_I, S_P^0, S_B^0), Temporal Scaling level (δ) and amounts of FEC $(S_{IF}, S_{PF}$,-$S_{BF})$. Another program was built to search through all combinations of FEC (S_{IF}, S_{PF}, S_{BF}) and Temporal Scaling levels (δ). Initially, each combination of FEC and scaling are tested to determine if this combination satisfies the TCP-Friendly rate constraint. If this combination does not satisfy the constraint, the search program goes to the next iteration. If the constraint is satisfied, the playable frame rate is estimated for this FEC and scaling combination. After searching all the combinations of FEC and scaling patterns within the constrained search space, the program produces the maximum playable frame rate, the adjusted FEC (S_{IF}, S_{PF}, S_{BF}), and the Temporal Scaling (δ) required to achieve this maximum rate.

Using these programs, the optimal playable frame rates over a range of network and application settings are explored. For each set of network and application parameters, the playable frame rates are compared for MPEG streaming without FEC, MPEG streaming with small fixed FEC (1/0/0), large fixed FEC (4/2/1) and MPEG streaming with adjusted FEC.

In all cases, the total bandwidth used by the MPEG video and FEC is temporal scaled to meet TCP-Friendly constraints using Pre-Encoding Temporal Scaling (PETS).

System Settings

The system parameter settings for the network and MPEG layers are the same as Table 4.5 in Section 4.3.1 (Analytical Experiments of MIPOTS). Specifically, the I frame has 25 packets, the P frame has 8 packets and the B frame has 3

packets. A commonly used MPEG GOP pattern, '`IBBPBBPBBPBB`', (GOP(3,8)) and a typical full motion frame rate R_F of 30 frames per second (fps) were used. The packet size s is 1KB, round-trip time t_{RTT} is 50 ms, and the packet loss probability p ranges from 0.01 to 0.04 in steps of 0.001.

Analysis

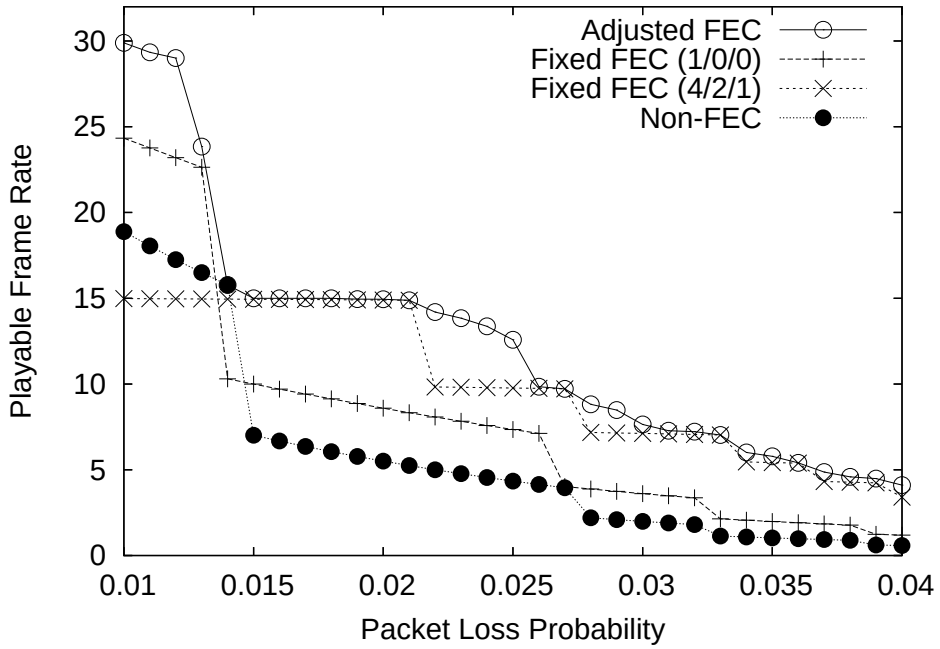


Figure 4.9: Comparison of Playable Frame Rates

Figure 4.9 depicts the playable frame rates for each of the four schemes. The x-axis is the packet loss probability, and the y-axis is the playable frame rate. In the figure, adjusted FEC provides the highest playable frame rate under all network and video conditions. The benefits of adjusted FEC over non-FEC are substantial, surpassing over 5 frames per second for most loss rates. When the loss rate is low ($\leq$ 1.5%), FEC(4/2/1) does worse with playable frames below the non-FEC scheme. This situation happens because the fixed number of FEC packets added is a larger fraction of overhead for

the video frames. When the loss rate is higher, FEC(4/2/1) does substantially better and provides close to the maximum frame rate achieved by adjusted FEC. FEC(1/0/0) does significantly worse since it does not provide enough protection. However, FEC(1/0/0) still outperforms the non-FEC scheme.

These figures show fixed FEC only works well for specific network and MPEG conditions. For example, FEC(1/0/0) works nearly as well as the adjusted FEC for low loss while FEC(4/2/1) works nearly as well as the adjusted FEC for high loss. However, when the network and MPEG conditions change, both fixed FEC patterns chosen are less effective than the more robust adjusted FEC scheme. This general behavior holds for other fixed FEC choices, regardless of the specific input patterns used.

Adjusting FEC

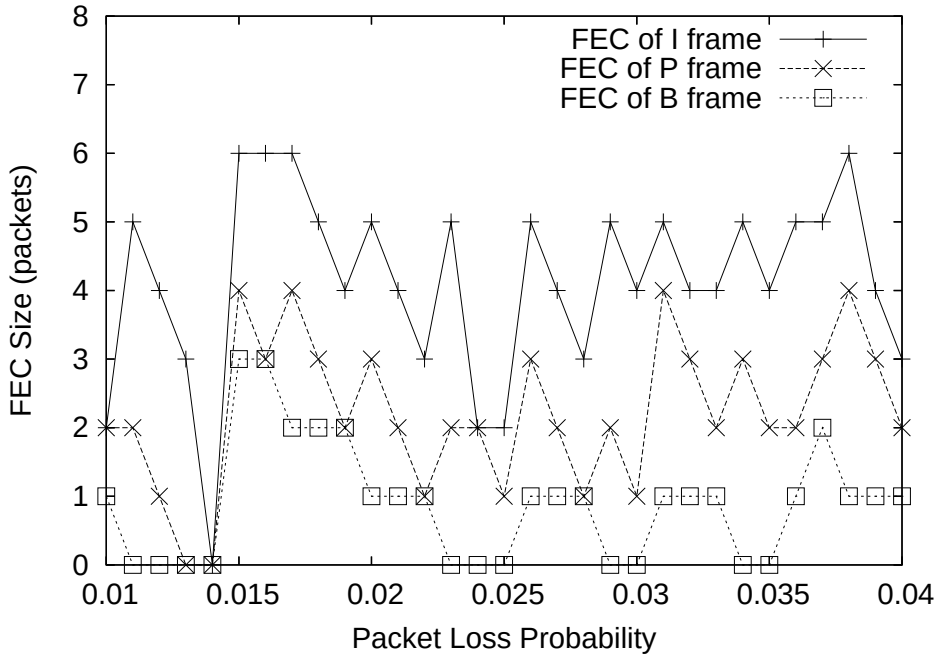


Figure 4.10: Adjusted FEC Pattern

To better explain the benefits of adjusted FEC presented in the previous

section, we now analyze how FEC is adjusted for various fixed loss rates.

Figure 4.10 gives the breakdown of the adjusted FEC for each I, P, and B frame that produces the maximum playable frame rate versus the loss probability. Generally, with adjusted FEC, the most important I frames have the highest transmission probability followed next by the P frames and lastly by the least important B frames.

Temporal Scaling Level

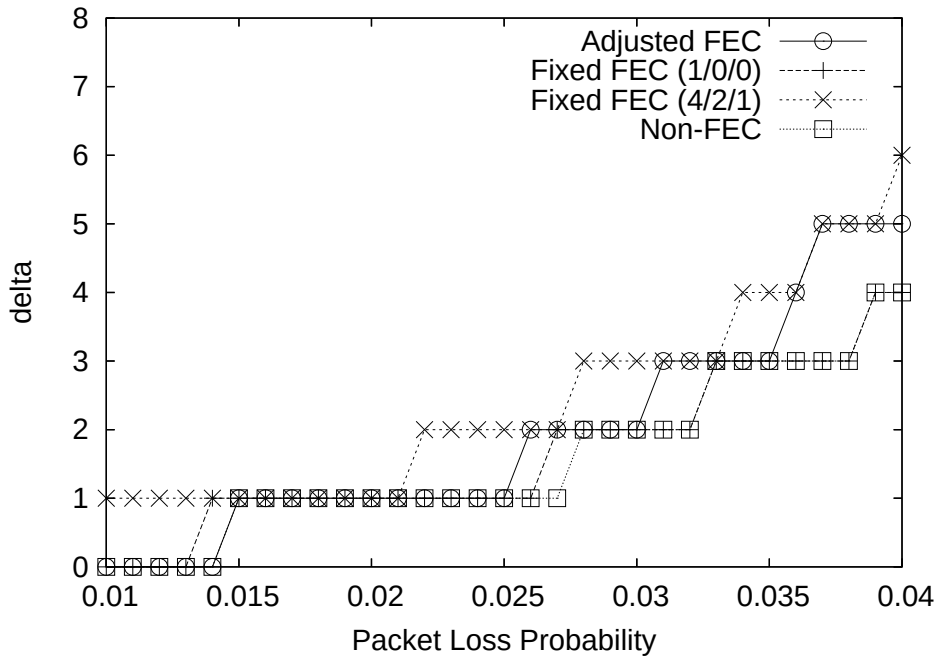


Figure 4.11: Temporal Scaling Level

Figure 4.11 depicts the Temporal Scaling level δ for each of the four FEC schemes. The x-axis is the packet loss probability, and the y-axis is δ. The figure shows δ increases with the loss probability. For the large fixed FEC, the overhead consumes more bandwidth so the scaling level is higher than other schemes. For the small fixed FEC or non-FEC, the overhead is small so less scaling is required. MPEG video with adjusted FEC must discard

slightly more frames than the same MPEG video without FEC. However, the additional packet space saved by the discards can be effectively used for FEC packets.

4.3.3 Summary

The analytic experiments presented indicate that adjusting FEC with Temporal Scaling provides an improvement over current approaches. The adjusted FEC mechanism always achieves a higher playable frame rate than MPEG video without FEC and provides a higher playable frame rate than any fixed FEC approaches when taken over a wide range of possible MPEG encoding and network conditions. The results also show, to improve the streaming quality, the most important I frames must have the highest protection followed next by the P frames and lastly by the least important B frames.

4.4 Media-Independent FEC and Quality Scaling (MIQS)

This section uses Quality Scaling to adjust the bitrate to the capacity constraint when Media-Independent FEC is used to recover packet loss.

In MPEG, the DCT coefficient of the video signal is quantized by dividing by an integer (the *quantization value* v_Q), and rounding to the nearest integer. When using higher quantization values, each MPEG frame is encoded with lower precision, and transmitted with fewer bits. Thus, this scaling technique reduces the bitrate of the streaming video.

This section focuses on using adaptive quantization values, but the model and algorithm developed is independent of the scaling technique and only requires the relationships between scaling level, encoding bitrate and video quality. See Section 2.4.4 for more details on Quality Scaling methods.

When Quality Scaling is used, it is not appropriate to assume each playable frame has the same quality, since each frame could have a different quality level. In this case, new quality measurements need to be used to evaluate the quality of the streamed video. In this work, the VQM metric (see Section 2.5.4) developed by the Institute for Telecommunication Sciences[3] is used as an objective video quality measurement tool.

4.4.1 System Layers

System Layer	Parameters/Variables
MPEG	$S_I, S_P, S_B, N_P, N_B, R_F$
ARMOR	$S_{IF}, S_{PF}, S_{BF}, v_Q$
Network	p, t_{RTT}, s, T

Table 4.7: System Layers and Parameters/Variables

[3] http://www.its.bldrdoc.gov/n3/video/vqmsoftware.htm

Similar to Section 4.1, the system layers and parameters/variables are shown in Table 4.7, where R_F, S_I, S_P, S_B, N_P, N_B, N_G, S_{IF}, S_{PF}, S_{BF}, p, t_{RTT}, s, T have the same meaning as Table 4.1 on Page 50 and the new ARMOR parameter is v_Q, the quantization value. While it is possible to use different quality values for the different frame types, it is difficult to model the quality dependencies among different frames, so I, P and B frames are assumed to have the same quantization value and the study with different quantization values is left as future work.

As in the previous section, it is assumed the network protocol provides loss rates, round-trip times and packet sizes, while the streaming video application provides details on the MPEG frame characteristics. With similar steps, a ARMOR quality model is developed to explore the effects of various choices of FEC and Quality Scaling on video performance.

4.4.2 Distortion from Quality Scaling

When a video is streamed over an unreliable network under a capacity constraint, its perceptual quality is degraded by two factors: quality scaling and frame loss. Scaling distortion, caused by a high quantization value, appears visually as coarse granularity in every frame. Frame loss due to network packet loss yields jerkiness in the video playout.

This study uses the Video Quality Model (VQM) [66], an objective video quality measurement, to approximate the distortion due to Quality Scaling. Section 4.4.3 uses playable frame rate to estimate the distortion from frame loss. Section 4.4.4 presents a new quality metric, *distorted playable frame rate*, that combines these two factors.

VQM takes an original video and a distorted video as input and returns a distortion value D between 0 (no distortion) and 1 (maximum distortion).

Previous research [27] implies that perceptual video distortion varies exponentially with the quantization value. Employing VQM to measure D in videos encoded with varying quantization levels, our preliminary studies (see Table 4.9 at Page 99) show that D can be approximated as an exponential function of the quantization value v_Q:

$$D = \hat{D} \cdot {v_Q}^{\lambda_D} \tag{4.26}$$

where v_Q is the quantization value, $\hat{D}$ is the VQM distortion when $v_Q = 1$, and λ_D is the exponential coefficient. Table 4.9 in Section 4.4.6 provides an example that shows how accurately this function fits real video data.

4.4.3 Playable Frame Rate

Frame Size

The compressed frame sizes change with the quantization value. Previous research [27, 79] demonstrates that MPEG streaming bitrate can be approximated by an exponential function of the quantization value v_Q. Our preliminary experiments (see Table 4.9 on Page 99) suggest frame size can be estimated by exponential functions of quantization value v_Q given as:

$$\begin{cases} S_I = \hat{S_I} \cdot {v_Q}^{\lambda_I} \\ S_P = \hat{S_P} \cdot {v_Q}^{\lambda_P} \\ S_B = \hat{S_B} \cdot {v_Q}^{\lambda_B} \end{cases} \tag{4.27}$$

where v_Q is the quantization value, $\hat{S_*}$ is the frame size when $v_Q = 1$, and λ_* is the exponential coefficient. Note, all the results S_* are rounded up to the nearest integer $\lceil S_* \rceil$ since video frames must be sent over the network in an integer number of packets. Table 4.9 on Page 99 shows how accurately these functions fit real video frame sizes.

Successful Frame Transmission Probability

Given I, P, and B frame sizes, and the distribution of redundant FEC packets added to each frame type, the Reed-Solomon Code equation (Equation 2.1) provides the probability of successful transmission for each frame type, knowing the amount of redundancy added by Media-Independent FEC:

$$\begin{cases} q_I = q(S_I + S_{IF}, S_I, p) \\ q_P = q(S_P + S_{PF}, S_P, p) \\ q_B = q(S_B + S_{BF}, S_B, p) \end{cases} \tag{4.28}$$

While this equation looks similar to Equation 4.1 in Section 4.1, the difference is S_I, S_P, S_B change as functions of v_Q here but they remain constant in Equation 4.1.

Playable Frame Rate

Similar to previous ARMOR models (MIPOTS in Equation 4.12 on Page 54 and MIPETS in Equation 4.24 on Page 59), the following function is used to estimate the total playable frame rate for streaming MPEG:

$$\begin{aligned} R &= R_I + R_P + R_B \\ &= G \cdot q_I \cdot (1 + \tfrac{q_P - q_P^{N_P+1}}{1-q_P} + N_{BP} \cdot q_B \\ &\qquad \cdot (\tfrac{q_P - q_P^{N_P+1}}{1-q_P} + q_I \cdot q_P^{N_P})) \end{aligned} \tag{4.29}$$

where G is the GOP rate, N_P is the number of P frames, and N_{BP} is the number of B frames between two reference frames.

4.4.4 Distorted Playable Frame Rate

Quality Scaling uses a higher quantization value to encode the video, causing intra-frame quality distortion. Frame loss lowers the playable frame rate and

is referred to as inter-frame quality distortion.

Since the inter-frame and intra-frame distortion components are independent, it is assumed they contribute independently to the overall distortion. Hence, quality distortion can be represented by a function of these two factors. To stream the highest quality video possible, the media server needs to use the best Quality Scaling level and the media client needs to receive all the frames. Thus, these two factors are combined into a multiplicative function, referred to as the distorted frame rate, R_D:

$$R_D = (1 - D) \cdot R \tag{4.30}$$

where D is the quality distortion from Equation 4.26 and R is the playable frame rate from Equation 4.29.

The motivation behind R_D is as follows. If a video is streamed with the best quantization value, its Quality Scaling distortion is 0 and video quality is determined only by the playable frame rate R. With any other quantization value, every frame carries less visual detail and its contribution to the video quality (measured by frame rate) is reduced by the quality distortion D. A preliminary user study (shown in Section 4.4.6) shows a correlation between user perceptual quality and distorted playable frame rate R_D. This suggests that R_D may be a reasonable representation of overall video quality. A more comprehensive user study (Chapter 5) shows more confident results that user perceptual quality can be accurately represented by distorted playable frame rate.

4.4.5 Optimization Algorithm

For given network conditions and MPEG video parameters, the total distorted playable frame rate R_D varies with quantization value and the amount of FEC for each frame type as a function $R_D(v_Q, (S_{IF}, S_{PF}, S_{BF}))$ where the streaming bitrate is limited by a capacity constraint, T. Thus, an optimization algorithm can use this model to maximize the distorted playable frame rate, R_D, using the following operation research equation:

$$\begin{cases} Maximize: \\ \quad R_D = (1 - D(v_Q)) \cdot R(v_Q, (S_{IF}, S_{PF}, S_{BF})) \\ Subject\ to: \\ \quad G \cdot ((S_I(v_Q) + S_{IF}) + N_P \cdot (S_P(v_Q) + S_{PF}) \\ \qquad + N_B \cdot (S_B(v_Q) + S_{BF})) \leq T \end{cases} \tag{4.31}$$

Similarly to previous ARMOR algorithms, given that the optimization problem is expressed in terms of integer variables over a restricted domain, a search of the discrete space is feasible. With fixed input values for (p, RTT, s), (G, N_P, N_B) and functions of ($S_I(v_Q)$,$S_P(v_Q)$,$S_B(v_Q)$), each set of values of ARMOR variables, v_Q and (S_{IF}, S_{PF}, S_{BF}), determines the distorted playable frame rate R_D using the following steps:

1. Approximate the video frame sizes (S_I,S_P and S_B) using v_Q in Equation 4.27.

2. Estimate total video streaming bitrate using the video frame sizes and the FEC frame sizes. If the estimated bitrate is larger than the capacity constraint T, the set of values of ARMOR variables are invalid and R_D is returned as 0.

3. Otherwise, use the video frame sizes and the FEC sizes to determine the

successful transmission probabilities (q_I, q_P and q_B) from Equation 4.28.

4. Estimate the playable frame rate R by inputting (q_I, q_P and q_B) into Equation 4.29.

5. Use v_Q in Equation 4.26 to approximate D.

6. Employ R and D in Equation 4.30 to estimate the distorted playable frame rate, R_D.

With these steps for each set of values, the space of possible variable values for v_Q and (S_{IF}, S_{PF}, S_{BF}) is exhaustively explored to determine the quantization value and the amount of FEC packets for each frame type that maximizes the distorted playable frame rate under the capacity constraint. Since the search can be done in real-time[4], the determination of optimal choices for adaptive FEC and Quality Scaling is feasible for streaming MPEG.

4.4.6 Analytical Experiments

Methodology

Using the optimization algorithm, the distorted playable frame rates over a range of network and application settings are explored. For each set of network and application parameters, the playable frame rates are compared for MPEG streaming with quality adjusted FEC, MPEG streaming with two types of fixed FEC, and MPEG streaming without FEC:

1. Adjusted FEC: Before transmission, the server employs the optimization algorithm based on the ARMOR-MIQS to determine the FEC and

[4]It takes about 100 milliseconds to find the best FEC and scaling pattern using our approach on a Pentium-3 800 MHz PC for a GOP of `IBBPBBPBBPBBPBB`. Optimizations of the code and a faster machine will allow searching to be done even faster.

Quality Scaling levels that maximize the distorted playable frame rate R_D and uses these for the entire video transmission.

2. Large Fixed FEC: The server protects each frame with 15% FEC packets (rounded up to the nearest integer). This FEC pattern provides strong protection to each frame and roughly represents the relative importance of the I, P and B frames [35, 47].

3. Small Fixed FEC: Each I frame receives 1 FEC packet. This simple FEC pattern protects the most important frame, the I frame. Protecting the I frame is a scheme used by other researchers [21, 75].

4. Non-FEC: No FEC is added to the video.

The total bitrate used by the MPEG video and FEC is scaled to meet a TCP-Friendly capacity constraint [61] using Quality Scaling.

System Settings

Network Layer		MPEG Layer	
t_{RTT}	50 ms	N_P	4 frames per GOP
s	1 Kbyte	N_B	10 frames per GOP
p	0.01 to 0.04	R_F	30 frames per second

Table 4.8: System Parameter Settings

Table 4.8 presents the system parameter settings for the network and application layers. A commonly-used MPEG GOP pattern, '`IBBPBBPBBPBBPBB`', and a typical full motion frame rate R_F of 30 frames per second (fps) are used. The packet size s, round-trip time t_{RTT} and packet loss probability p are chosen based on the characteristics of many network connections [15, 38]. For all experiments, the parameters are fixed, except for packet loss probability p, which was varied from 0.01 to 0.04 in steps of 0.002.

Two picture sequences are used. The first video, *Paris*, from PictureTel, shows two people sitting at a table and talking while making high-motion gestures (see Figure 4.1 on Page 64). It has 900 raw images and lasts for 30 seconds, providing a frame rate of 30 fps. The image size is 352x288 pixels (CIF). The Berkeley MPEG encoder *mpeg_encode* [59] is used to encode the images with different quantization values as one-minute long videos. From the output videos, the frame sizes are extracted with the Berkeley MPEG statistics tool *mpeg_stat* [59] and the quality distortion is extracted with VQM. Statistical analysis software *SPSS* [5] is used to fit the relation between quality distortion and v_Q to a function as in Equation 4.26, and the relation between frame sizes and v_Q in Equation 4.27. The equations then become:

$$\begin{cases} D = 0.025 \cdot {v_Q}^{0.87} \\ S_I = 81.51 \cdot {v_Q}^{-0.70} \\ S_P = 52.94 \cdot {v_Q}^{-1.21} \\ S_B = 15.47 \cdot {v_Q}^{-0.79} \end{cases} \tag{4.32}$$

Table 4.9 shows how these analytical functions fit the Paris data with some representative quantization values. In the table, v_Q is the quantization value, D', S'_I, S'_P, S'_B are estimated by the analytical functions, and D, S_I, S_P, S_B are the real values from the video analysis. Overall, the functions fit the data well.

The second video sequence is *Tennis*, which comes with the Berkeley tools [59]. *Tennis*, a short clip of two men playing ping-pong, has 150 raw images and lasts for 5 seconds, providing a frame rate of 30 fps. The size of each frame is 352x240 pixels. Again, the Berkeley MPEG tools, VQM and SPSS are used to approximate the quality distortion and frame sizes to functions of Quality

[5] http://www.spss.com

v_Q	$D^{'}$	D	$S_I^{'}$	S_I	$S_P^{'}$	S_P	$S_B^{'}$	S_B
5	0.10	0.09	26.4	26.5	7.5	7.8	4.3	4.5
8	0.15	0.15	19.0	19.5	4.2	4.6	2.9	2.8
12	0.21	0.22	14.3	14.5	2.6	2.8	2.1	2.0
18	0.31	0.33	10.7	10.7	1.6	1.6	1.5	1.4
24	0.39	0.38	8.8	8.7	1.1	1.1	1.2	1.2
31	0.49	0.48	7.3	7.2	0.8	0.7	1.0	1.1

Table 4.9: Estimated Value by Equation 4.32 versus Real Video Data

Scaling level:

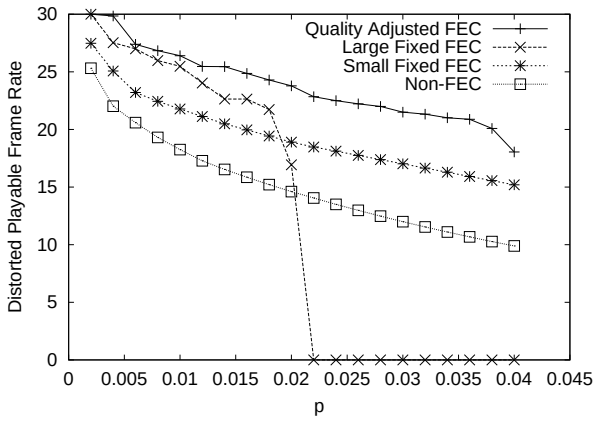
$$\begin{cases} D = 0.041 \cdot {v_Q}^{0.69} \\ S_I = 74.55 \cdot {v_Q}^{-0.86} \\ S_P = 96.22 \cdot {v_Q}^{-1.31} \\ S_B = 33.27 \cdot {v_Q}^{-1.01} \end{cases} \tag{4.33}$$

The functional fit for *Tennis* is not shown, but the approximation errors are less than 5%.

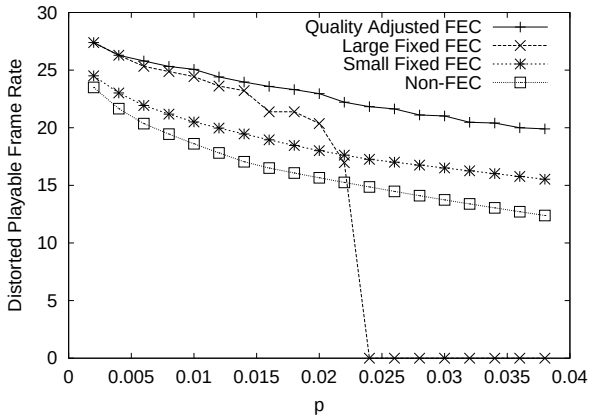
Note, analyzing videos and fitting the results to these exponential functions is a time-intensive operation. It may be possible to analyze a large variety of video types and find parameters for the exponential functions, based either on the frames sizes, frame rates and video content, that are generally effective. This analysis is left as future work.

Analysis of Results

The *Paris* and *Tennis* videos are used to approximate their Quality Scaling distortion D and frame sizes to functions as Equation 4.26 and Equation 4.27 and instantiate them to Equation 4.32 and Equation 4.33. These functions and the optimization algorithm are used to search the FEC and Quality Scaling values to maximize the distorted playable frame rate for the four approaches in Section 4.4.6.



a. Paris Video



b. Tennis Video

Figure 4.12: Distorted Playable Frame Rates

Figure 4.12 graphs the distorted playable frame rates for the four FEC choices for the *Paris* and *Tennis* videos. The x-axes are the packet loss probabilities, and the y-axes are the distorted playable frame rates. From the data in these figures, quality adjusted FEC provides the best quality under all network and video conditions. The benefits of quality adjusted FEC over non-FEC are substantial, with quality adjusted FEC providing 5-10 more frames per second for all rates. The small fixed FEC approaches usually improve playable frame rates over non-FEC video, especially when loss rates are high. However, the small fixed FEC frame rates are still much lower than the frame rates with quality adjusted FEC. Large FEC achieves the playable frame rate provided

by quality adjusted FEC for low loss rates because the TCP-Friendly rate is relatively high. With limited capacity (at high loss rates), the Quality Scaling level is high (> 16), and the large FEC overhead becomes significant. There is little reduction in the frame sizes as the scaling level increases (to a maximum of 31) and Quality Scaling is unable to conserve enough capacity to overcome the effect of the large FEC overhead. The result is that none of the original video data is sent. These trends hold for both videos despite differences in the content between the two clips.

A preliminary user study was conducted to compare the impact of FEC choices on actual users and to ascertain the efficacy of the distorted playable frame rate measure, R_D, as a measure of perceived visual quality. Four versions of the *Paris* video clips were generated, simulating the Quality Adjusted FEC approach, the two fixed FEC strategies and the Non-FEC approach on a network with 0.02 packet loss and a TCP-Friendly constraint of 1.17 Mbps. No local concealment technique is used at the receiver side; if a frame was not playable, the preceding playable frame was repeated. Ten undergraduate students were asked to rate the quality from 0 (worst) to 10 (best) of the four videos using the original videos with no packet loss as a reference. The ratings were provided after viewing each clip twice in a different random order by each student.

Repair method	D	R	R_D	Q
Quality Adjusted FEC	0.17	28.55	23.78	6.89
Small Fixed FEC	0.20	23.58	18.90	4.44
Large Fixed FEC	0.44	30.00	16.93	3.89
Non-FEC	0.28	20.17	14.61	3.50

Table 4.10: Preliminary User Study

Table 4.10 displays the average user quality rating Q compared with the VQM distortion D, the playable frame rate R, and the distorted playable

frame rate R_D for the four videos. The results indicate that the perceived qualities of the videos with FEC are significantly higher than the videos without FEC. Additionally, QAFEC videos appear noticeably better under all the conditions than videos with fixed FEC. Perhaps most importantly, the correlation between Q and R_D suggests that the distorted playable frame rate R_D appropriately represents perceived quality, accounting for both the temporal aspects of the video that influence perceived quality (R) and the quality aspects of the video that influence perceived quality (D). Later, in Chapter 5, a more comprehensive user study shows more experimental validation that R_D can be used to represent user's perceptual quality very well.

It may be surprising that non-FEC has a higher VQM distortion D since non-FEC has a higher encoding bitrate without any FEC overhead. The reason is, if non-FEC chooses a better (lower) quantization level v_Q under the capacity constraint, it will increase the frame size, reducing the successful transmission probability for each frame and getting a much lower playable frame rate. So, non-FEC selects a conservative quantization level to get better overall quality.

4.4.7 Summary

This investigation studies adjusting FEC with Quality Scaling for streaming MPEG. An analytic model is proposed that captures the quality distortion of streaming MPEG in the presence of Quality Scaling and frame loss. Using this model, an optimization algorithm determines the optimal adjustment of FEC and Quality Scaling under a capacity bound, accounting for both the network conditions and video parameters.

The analytic experiments show that adjusted FEC has significant advantages. Quality adjusted FEC always achieves higher quality than MPEG video without FEC and provides higher video quality than fixed FEC approaches

when taken over a range of MPEG encoding and network conditions. A preliminary user study confirms the experiments in demonstrating that the perceived video quality with quality adjusting FEC are significantly better than the videos with fixed FEC or without FEC. The user study also suggests that the proposed distorted playable frame rate, R_D, has the appropriate trend in capturing the distortion to video quality from Quality Scaling and frame loss. A more comprehensive user study is presented in Chapter 5.

4.5 Media Independent FEC with Temporal and Quality Scaling (MITQS)

Temporal Scaling and Quality Scaling are commonly used to scale back real-time streaming video data rates to adjust to a capacity constraint, caused by the Internet Service Provider (ISP)'s negotiated rate, or to be TCP-Friendly [25]. While many researchers have studied Temporal Scaling or Quality Scaling [27, 41, 47, 87, 88] and many commercial video streaming products have incorporated these scaling methods, to the best of our knowledge there has been no systematic study of the combination of Temporal Scaling and Quality Scaling.

The previous sections focused on the impact of FEC on video streaming, using first Temporal Scaling (Section 4.1) and then Quality Scaling (Section 4.4), to stream under a capacity constraint. Our results showed that FEC is critical for acceptable performance and works best when dynamically adjusted to the current network packet loss rate and capacity constraint. Either Temporal or Quality Scaling provides acceptable methods of scaling, but the research made no attempt at comparing, much less combining, the two scaling approaches.

This section studies the combination of both Temporal and Quality scaling. Specifically, our previously developed analytic model is extended to characterize the performance of MPEG video with the combination of Temporal Scaling and Quality Scaling. This new model incorporates both a Temporal and Quality Scaling level and adjusts the number of FEC packets for each MPEG frame type. From the MPEG characteristics, video distortion is approximated using the scaling parameters and the video frame loss rate. Then, a new optimization algorithm is built to exhaustively search all possible combinations of scaling levels and FEC patterns to find the configuration that yields the best video quality under the capacity constraint.

4.5.1 Background

Quality Scaling

This study assumes the SPEG (Scalable MPEG) [41] Quality Scaling model, with every DCT coefficient divided into four layers: one base layer and three advance layers. A DCT coefficient, C, is partitioned into the layers using the following equations:

$$\begin{aligned} &Base\ Layer\ L0: && C_0 = C >> 3 \\ &1st\ Advance\ Layer\ L1: && C_1 = (C >> 2)\&1 \\ &2nd\ Advance\ Layer\ L2: && C_2 = (C >> 1)\&1 \\ &3rd\ Advance\ Layer\ L3: && C_3 = C\&1 \end{aligned} \tag{4.34}$$

At the receiver/player side, the above steps are reversed to reconstruct the original MPEG video where zero is used instead when some advance layer(s) is (are) absent. This is analogous to using a high quantization value during MPEG encoding. Assuming the highest quantization value used is 3 (this yields a high fidelity quality and reasonable bitrate), it is not difficult to define the relationship of the Quality Scaling level l_{QS}, transmitting SPEG layers and equivalent quantization value v_Q as in Table 4.11. Since SPEG needs to use extra header information to indicate layer information (a 15%-25% overhead in [41]), 20% overhead is used.

Scaling Level (l_{QS})	SPEG Layers	Equ. Quan. Val. v_Q
0	L0+L1+L2+L3	3
1	L0+L1+L2	6
2	L0+L1	12
3	L0	24

Table 4.11: Quality Scaling Levels

Temporal Scaling

We use POst-encoding Temporal Scaling (POTS) (see Section 2.4.1) as the Temporal Scaling method. We use only 4 Temporal Scaling levels to be comparable to SPEG Quality Scaling. Table 4.12 lists the 4 scaling levels, accounting for the MPEG frame dependencies and minimizing the effect of Temporal Scaling on the quality of the received video. In the table, N_{PD} and N_{BD} are defined as the number of P or B frames which will be transmitted in one GOP, respectively, with the scaling patterns provided for each scaling level, l_{TS}. Since typical MPEG decoders detect, and accommodate, lost frames, the frames selected for discarding can be removed at the sender with effectively no additional overhead.

Scaling Level (l_{TS})	N_{PD}	N_{BD}	Scaling Pattern
0	4	10	`IBBPBBPBBPBBPBB`
1	4	5	`IB-PB-PB-PB-PB-`
2	4	0	`I--P--P--P--P--`
3	0	0	`I--------------`

Table 4.12: Temporal Scaling Levels

4.5.2 System Layers

Layer	Parameters/Variables
MPEG	G, S_I, S_P, S_B
ARMOR	$l_{TS}, N_{PD}, N_{BD}, l_{QS}, v_Q, S_{IF}, S_{PF}, S_{BF}$
Network	p, t_{RTT}, s, T

Table 4.13: System Layers and Parameters/Variables

Similar to Section 4.1 and Section 4.4, the system layers and parameters/variables are indicated in Table 4.13, where S_I, S_P, S_B, N_{PD}, N_{BD}, v_Q, S_{IF}, S_{PF}, S_{BF}, p, t_{RTT}, s, T have the same meaning as Table 4.1 and Table 4.7 and the new ARMOR variables are:

l_{TS}: the Temporal Scaling level, as in Table 4.12.

l_{QS}: the Quality Scaling level, as in Table 4.11.

4.5.3 Distortion from Quality Scaling

As discussed in Section 4.4, when a video is streamed over an unreliable network under a capacity constraint, its perceptual quality can be degraded by two factors: quantization and frame loss. Frame loss, caused by Temporal Scaling and network packet loss, appears visually as jerkiness in the video playout. The quantization distortion is caused by low accuracy of the DCT coefficients and appears visually as coarse granularity in every frame. The video quality distortion, D, can be approximated by an exponential function of the quantization value v_Q as:

$$D = \hat{D} \cdot {v_Q}^{\lambda_D} \tag{4.35}$$

where v_Q is the quantization value now decided by l_{QS} as in Table 4.11, $\hat{D}$ is the VQM distortion when $v_Q = 1$, and λ_D is the exponential coefficient. The estimated VQM distortion is then used to measure the quality distortion from Quality Scaling, as indicated in Section 4.5.5.

4.5.4 Playable Frame Rate

Frame Size

Similar to Section 4.4 (MIQS), when the quantization values change, the frame sizes change and the frame sizes can be estimated by an exponential function

of quantization value v_Q, given as:

$$\begin{cases} S_I = \hat{S_I} \cdot {v_Q}^{\lambda_I} \\ S_P = \hat{S_P} \cdot {v_Q}^{\lambda_P} \\ S_B = \hat{S_B} \cdot {v_Q}^{\lambda_B} \end{cases} \tag{4.36}$$

where v_Q is the quantization value now decided by l_{QS} as in Table 4.11, $\hat{S_*}$ is the frame size when $v_Q = 1$, and λ_* is the exponential coefficient. Note, all the results S_* need to be rounded up to the nearest integer $\lceil S_* \rceil$ since each video frame must be divided into a whole number of packets when sent on the network.

Playable Frame Rate

Section 4.1.4 (MIPOTS) derived a model to estimate total playable frame rate for streaming MPEG with Temporal Scaling. With the model, the total playable frame rate R is:

$$R = R(p, (N_{PD}, N_{BD}), (S_I, S_P, S_B), (S_{IF}, S_{PF}, S_{BF})) \tag{4.37}$$

Since N_{PD} and N_{BD} are decided by l_{TS} as in Table 4.12, and S_I, S_P, and S_B are decided by l_{QS} as in Equation 4.36 and Table 4.11, this equation can be written as:

$$R = R(p, l_{TS}, l_{QS}, (S_{IF}, S_{PF}, S_{BF})) \tag{4.38}$$

The estimated frame rate is then used to measure the quality distortion from frame loss, as indicated in Section 4.5.5.

4.5.5 Distorted Playable Frame Rate

We still use the distorted frame rate, R_D, from Section 4.4 (MIQS) to capture the inter-frame and intra-frame components of distortion in a multiplicative function:

$$R_D = (1 - D) \cdot R \tag{4.39}$$

where D is the quality distortion from Equation 4.35 and R is the playable frame rate from Equation 4.38.

Again, a comprehensive user study (Chapter 5) suggests that user perceptual quality can be accurately represented by distorted playable frame rate.

4.5.6 Optimization Algorithm

For given network conditions and MPEG video parameters, the total distorted playable frame rate R_D varies with the Quality Scaling level, the Temporal Scaling level, and the amount of FEC for each type of frame as a function $R_D(p, l_{TS}, l_{QS}, (S_{IF}, S_{PF}, S_{BF}))$ where the streaming bitrate is limited by the capacity constraint, T. Thus, this model can be used to optimize the distorted playable frame rate, R_D, using the following *operations research* equation:

$$\begin{cases} Maximize: \\ \quad R_D = (1 - D(v_Q)) \cdot R(p, l_{TS}, l_{QS}, (S_{IF}, S_{PF}, S_{BF})) \\ Subject\ to: \\ \quad G \cdot ((S_I(l_{QS}) + S_{IF}) + N_{PD}(l_{TS}) \cdot (S_P(l_{QS}) + S_{PF}) \\ \qquad + N_{BD}(l_{TS}) \cdot (S_B(l_{QS}) + S_{BF})) \leq T \end{cases} \tag{4.40}$$

Similar to previous ARMOR algorithms, given that the optimization prob-

lem is expressed in terms of integer variables over a restricted domain, a search of the discrete space is feasible. With fixed input values for (p, RTT, s), G and functions of $(N_{PD}(l_{TS}), N_{BD}(l_{TS})$ and $(S_I(l_{QS})$,$S_P(l_{QS})$,$S_B(l_{QS}))$, each set of values of ARMOR variables, l_{TS}, l_{QS}, and (S_{IF}, S_{PF}, S_{BF}), can determine the distorted playable frame rate R_D with the following steps:

1. l_{QS} is used to obtain a quantization value v_Q from Table 4.11. The video frame sizes (S_I,S_P and S_B) are then approximated using Equation 4.36.

2. The video streaming bitrate is estimated using the video frame sizes, the FEC frame sizes and (N_{PD}, N_{BD}). If the estimated bitrate is larger than the capacity constraint T, the set of values of ARMOR variables are invalid and R_D is returned as 0.

3. Otherwise, the playable frame rate R is estimated by inputting (p, l_{TS}, l_{QS}, S_{IF}, S_{PF}, S_{BF}) into Equation 4.37.

4. Using v_Q, the distortion from Quality Scaling D is approximated using Equation 4.35.

5. Knowing R and D, the distorted playable frame rate R_D is estimated using Equation 4.39.

With these steps for each set of values, the space of possible variable values for l_{TS}, l_{QS} and (S_{IF}, S_{PF}, S_{BF}) can be exhaustively searched to determine the scaling levels and the amount of FEC packets for each frame type to maximize the distorted playable frame rate under the capacity constraint. In fact, the computation required by the search can be done in real-time, making the determination of optimal choices for adaptive FEC feasible for most streaming MPEG connections.

4.5.7 Analytical Experiments

Methodology

Using the optimization algorithm (Equation 4.40) presented in Section 4.5.6, the distorted playable frame rates over a range of network and application settings are explored. For each set of network and application parameters, the distorted playable frame rates are compared for MPEG streaming with Temporal Scaling, Quality Scaling and the combination of them.

The MPEG streams with scaling are protected by one of four different FEC methods similar to previous sections:

1. Adjusted FEC: Before transmitting, the sender employs the optimization algorithm to determine the FEC and scaling levels that maximize the distorted playable frame rate R_D and uses these for the entire video transmission.

2. Large Fixed FEC: The sender protects each frame with FEC packets equivalent to 15% of the original frame size (rounded up to the nearest integer).

3. Small Fixed FEC: Each I frame receives 1 FEC packet.

4. Non-FEC: The sender adds no FEC to the video.

System Settings

A commonly-used MPEG GOP pattern, '`IBBPBBPBBPBBPBB`', and a typical full motion frame rate of 30 frames per second (fps) are used. The packet size s of 1 KB, round-trip time t_{RTT} of 50ms and packet loss probability p, which ranges from 0.005 to 0.08 in steps of 0.005, are chosen based on the characteristics of many network connections [15, 38].

We use the nine video clips from Section 4.2. Each video clip has 300 raw images and the size of each frame is 352x288 pixels (CIF). Their information can be found in Table 4.2 on Page 65 and Figure 4.1 on Page 64.

Comparison of Scaling methods

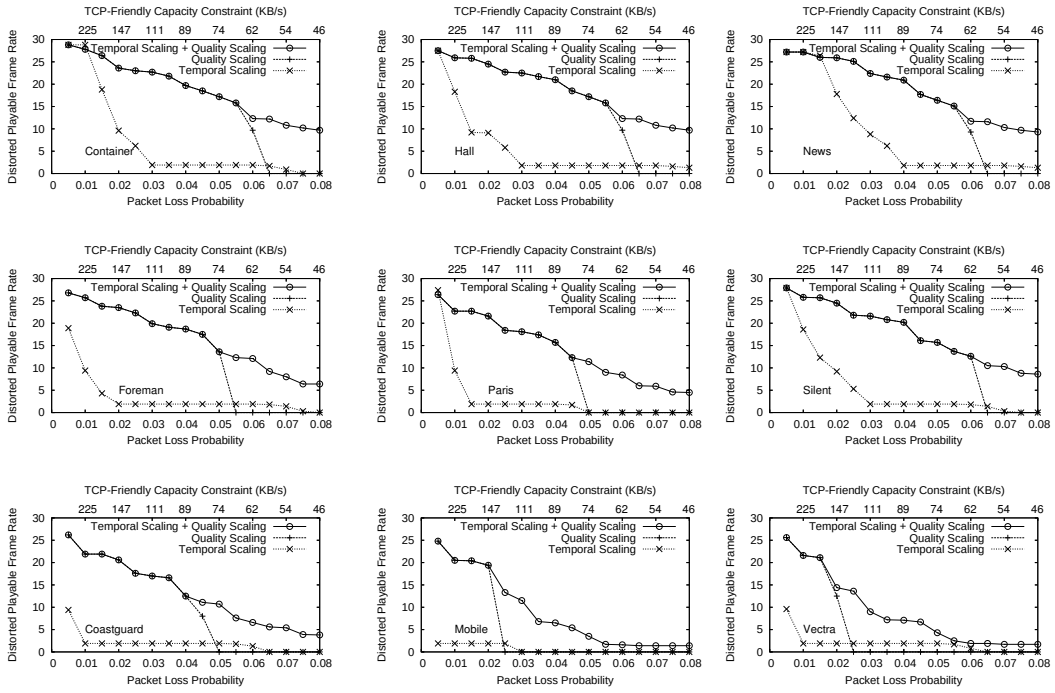


Figure 4.13: Comparison of scaling methods, with the 1st row of low motion clips, the 2nd row of medium motion clips and the 3rd row of high motion clips

Figure 4.13 depicts the distorted playable frame rates for the three scaling methods with Adjusted FEC for the nine videos. The bottom of each graph has an x-axis for the packet loss probabilities, the top of each graph has another x-axis for the corresponding TCP-Friendly capacity constraints, and the y-axes are the distorted playable frame rates. From the data in these figures, the combination of Temporal and Quality Scaling provides the best quality under all network and video conditions. When the packet loss is low and capacity limit is high, Quality Scaling provides performance nearly the same

as the combination of Temporal and Quality scaling. However, when the loss rate is high and the capacity is limited, Quality Scaling alone cannot scale enough to reduce the streaming bitrate below the capacity constraint and must be used with Temporal Scaling to yield reasonable video quality. These trends hold for all videos despite the differences in content among the clips. However, the motion properties of video clips are correlated to the differences between Temporal Scaling and Quality Scaling. For the high motion videos (*Coastguard*, *Mobile* and *Vectra*), Quality Scaling is always much better than Temporal Scaling. For the low motion videos (*Container*, *Hall* and *News*), Temporal Scaling can also provide reasonably high quality when the loss rate is low. The video motion also decides the point at which Temporal Scaling should be combined with Quality Scaling to provide a reasonable video quality, with high-motion videos needing the combination for lower capacity constraints than do low-motion videos.

Comparison of FEC Methods

Figure 4.14 depicts the distorted playable frame rates for the four FEC choices with the combination of Temporal Scaling and Quality Scaling for the nine videos. The bottom of each graph has an x-axis for the packet loss probabilities, the top of each graph has another x-axis for the corresponding TCP-Friendly capacity constraints, and the y-axes are the distorted playable frame rates. From the data in these figures, Adjusted FEC provides the best quality under all network and video combinations. The benefits in quality for Adjusted FEC over Non-FEC are substantial, with Adjusted FEC providing 5-10 more frames per second for all loss rates. The Small Fixed FEC approach usually improves playable frame rates over Non-FEC, especially when loss rates are high. However, Small Fixed FEC yields frame rates that are still much

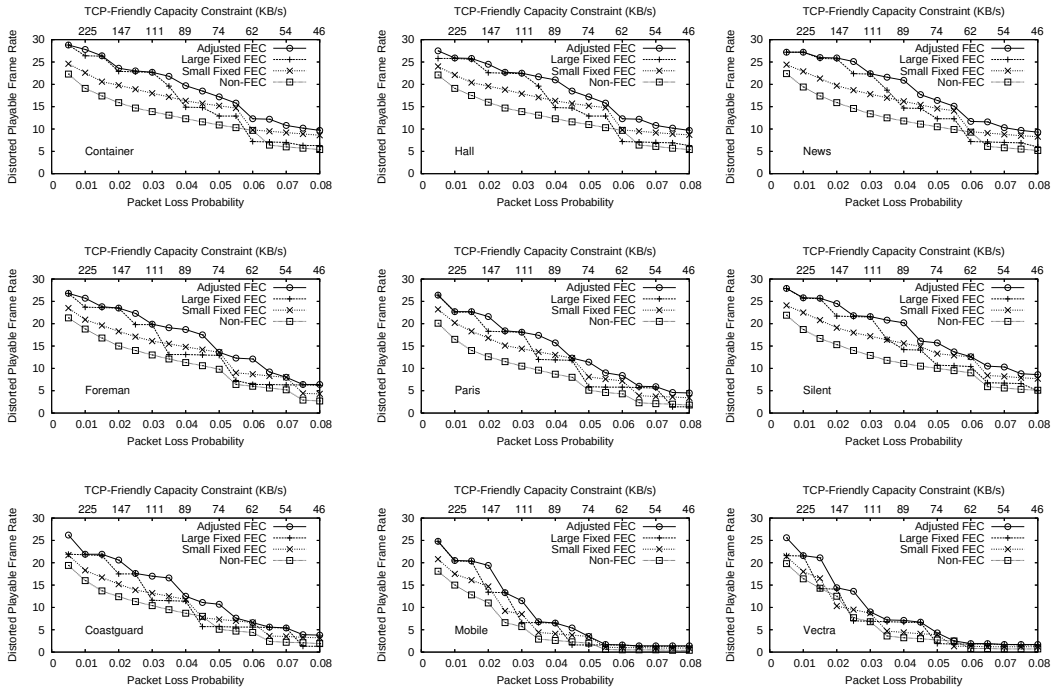


Figure 4.14: Comparison of FEC methods, with the 1st row of low motion clips, the 2nd row of medium motion clips and the 3rd row of high motion clips

lower than the frame rates with Adjusted FEC. The Large Fixed FEC approach achieves the playable frame rate provided by Adjusted FEC at low loss rates since the TCP-Friendly bitrate is relatively high. However, when the capacity is limited, Large Fixed FEC requires too much overhead and results in less video data being sent. These trends hold for all the videos despite the differences in motion content among the clips. The motion properties of video clips are correlated to the differences among the four FEC methods. For high motion clips, since the capacity constraints are tight, adjusted FEC can not improve the video quality as much as it does with low motion clips.

4.5.8 Summary

This section systematically compares Temporal Scaling, Quality Scaling and their combination, for streaming MPEG videos over a range of network and

video content conditions. An analytic model is proposed that captures the quality distortion of streaming MPEG in the presences of Temporal Scaling, Quality Scaling and repair with Forward Error Correction, as well as network conditions and video parameters. Using this model with an optimization algorithm determines the optimal adjustment of FEC and scaling given a capacity bound.

Analytic experiments on nine videos with varied motion characteristics show that when capacity constraints are moderate and loss rates are low, Temporal Scaling adds very little to the quality produced by using only Quality Scaling. When bitrates are low and loss rates are high, the combination of Quality Scaling and Temporal Scaling can still provide reasonable video quality. Additionally, the results imply that Quality Scaling provides better quality video than Temporal Scaling and that differences in their performance is correlated to video motion characteristics. Under conditions with loss, Adjusted FEC always achieves higher quality than MPEG video without FEC or any Fixed FEC approach.

4.6 Media Dependent FEC and Quality Scaling (MDQS)

In this section, Media-Dependent FEC is used to recover from packet loss, and Quality Scaling is used to adjust the bitrate to the constraint. Since Media Dependent FEC has the ability to partially recover a lost frame, a new video quality metric is introduced to estimate the quality of every frame separately and sums them up as the MDQS quality prediction. The MDQS model is then used in an optimization algorithm to maximize the video streaming quality. Later, Media Independent FEC with Quality Scaling (MIQS) is revisited with this new quality metric and is compared with Media Dependent FEC with Quality Scaling (MDQS).

4.6.1 System Introduction

System Layer	Parameters/Variables
MPEG	$S_I, S_P, S_B, N_P, N_B, R_F$
ARMOR	v_Q, v_{QF}
Network	p, t_{RTT}, s

Table 4.14: System Layers and Parameters/Variables

Similar to Section 4.1 and Section 4.4, the system layers and parameters/variables are indicated in Table 4.14, where R_F, S_I, S_P, S_B, N_P, N_B, N_G, l, p, t_{RTT}, s have been introduced in Table 4.1 and Table 4.7 and the new ARMOR variable is:

v_{QF}: the quantization value of the media-dependent FEC frames, assuming I, P and B frames have the same level.

Similar to Section 4.1 - Section 4.5, it is assumed the network protocol provides loss rates, round-trip times and packet sizes, while the streaming video application provides details on the MPEG frame characteristics. With

similar steps, a model and an algorithm are developed to explore the effects that FEC and Quality Scaling have on streaming video performance.

4.6.2 Frame Size

It was discussed that the frame size could be approximated by an exponential function of the quantization value as Equation 4.27. Moreover, since the redundant frames are just lower quality versions of MPEG frames when media-dependent FEC is in use, this equation is also applicable:

$$S_{*F} = \alpha_* \cdot {v_{QF}}^{\beta_*} \tag{4.41}$$

where * should be replaced by I, P or B, as appropriate.

4.6.3 Successful Frame Transmission Probabilities

Given sizes of I, P, and B frames and the FEC frame I_F, P_F and B_F, the probability of successful transmission for each frame type is:

$$q_* = q(S_*, 0, p) = (1-p)^{S_*} \tag{4.42}$$

where * should be replaced by I, P, B or I_F, P_F, B_F, as appropriate.

To aid in the discussion, a new term u_* is introduced to depict the unsuccessful frame transmission probability. For instance, u_I is the probability that an I frame is not decodable at the receiver. From this definition, u_* is equal to $1 - q_*$.

4.6.4 Weighted Playable Frame Rate

Similar to Media Independent FEC with Quality Scaling, a video quality measure, the Weighted Playable Frame Rate (WPFR), is introduced based on VQM. For each frame, which can be played at the receiver side, its contribution to the quality is only weighted by its quality distortion:

$$R_W = \sum_{*}(1 - D*) \tag{4.43}$$

where $*$ means every playable frame, which could be a successfully transmitted original MPEG frame or a lower quality FEC frame. Later, in Section 4.6.6, an analytical derivation will show that the weighted playable frame rate is a more general form of the distorted playable frame rate R_D.

WPFR of I frame

With Media-Dependent FEC, to transmit an I frame, an original I frame is sent to the receiver as well as a lower quality FEC frame. If the receiver gets the I frame, it plays it, if not, it tries to play the lower quality version. If the receiver can not receive any of these frames, it plays nothing. The probability that an original I frame could be played at the receiver is $H_I = q_I$, where "H" stands for "High Quality". The probability that a lower quality I frame is played in the receiver side is $L_I = (1 - q_I) * q_{IF} = u_I * q_{IF}$, where "L" stands for "Low Quality".

Taking these two parts together, an estimate of the WPFR of I frames is:

$$R_{WI} = G * ((1 - D(v_Q)) * H_I + (1 - D(v_{QF})) * L_I) \tag{4.44}$$

WPFR of P frames

For every P frame, P_i, can only be displayed when its preceding I or P frame is playable, with a high quality or a low quality, and itself or its low quality frame is successfully received.

If the referencing frame can be played at the higher quality and the current frame is also received with the higher quality, the current frame can be played at the higher quality. The probability of this case is:

$$H_{P_i} = H_{P_{i-1}} * q_P \tag{4.45}$$

If the referencing frame is the higher quality, but the current frame only receives the low quality frame, the current frame is decoded using the lower quality version. The probability of this case is:

$$L^1_{P_i} = H_{P_{i-1}} * (1 - q_P) * q_{PF} = H_{P_{i-1}} * u_P * q_{PF} \tag{4.46}$$

If the referencing frame is the lower quality, the current frame is played using the low quality version no matter if the original P frame or the low quality P frame is received. The probability of this case is:

$$L^2_{P_i} = L_{P_{i-1}} * (1 - u_P * u_{PF}) \tag{4.47}$$

Thus, the probability of displaying a lower quality version is:

$$L_{P_i} = L^1_{P_i} + L^2_{P_i} \tag{4.48}$$

and the estimated WPFR of the ith P frame is:

$$R_{WP_i} = G * ((1 - D(v_Q)) * H_{P_i} + (1 - D(v_{QF})) * L_{P_i}) \tag{4.49}$$

WPFR of B frames

All N_{BP} adjacent B frames have the same dependency relationship (they depend upon the previous and subsequent I or P frame) and thus these B frames all have the same playable rate.

For a B frame that precedes a P frame, say P_1, the B frame depends only on P_1. It is not necessary to consider the I or P frames before this B frame since these dependency relationships have already been accounted for in P_1. Thus:

$$\begin{aligned}
H_{B_{ij}} &= H_{P_{i+1}} * q_B \\
L^1_{B_{ij}} &= H_{P_{i+1}} * u_B * q_{BF} \\
L^2_{B_{ij}} &= L_{P_{i+1}} * (1 - u_B * u_{BF}) \\
L_{B_{ij}} &= L^1_{B_{ij}} + L^2_{B_{ij}} \\
R_{WB_{ij}} &= G * ((1 - D(v_Q)) * H_{B_{ij}} + (1 - D(v_{QF})) * L_{B_{ij}})
\end{aligned} \tag{4.50}$$

Total Weighted Playable Frame Rate

The total weighted playable frame rate is the sum of the playable frame rate for each frame type:

$$R_W = R_{WI} + R_{WP} + R_{WB} = R_{WI} + \sum_i R_{WP_i} + \sum_{i,j} R_{WB_{ij}} \tag{4.51}$$

For instance, when MDFEC is not used, using the above equations for

R_{WI}, R_{WP} and R_{WB}, the total playable frame rate, R_W, is:

$$\begin{aligned} R_W &= (1 - D(v_Q)) * (G \cdot q_I + G \cdot q_I . \frac{q_P - q_P^{N_P+1}}{1-q_P} \\ &\quad + N_{BP} \cdot G \cdot q_I \cdot q_B \cdot (\frac{q_P - q_P^{N_P+1}}{1-q_P} + q_I \cdot q_P^{N_P})) \\ &= (1 - D(l)) * G \cdot q_I \cdot (1 + \frac{q_P - q_P^{N_P+1}}{1-q_P} + N_{BP} \cdot q_B \\ &\quad \cdot (\frac{q_P - q_P^{N_P+1}}{1-q_P} + q_I \cdot q_P^{N_P})) \end{aligned} \tag{4.52}$$

4.6.5 Optimal Weighted Playable Frame Rate

For given network and video conditions, the total weighted playable frame rate R_W varies with the quality scaling pattern and the amount of FEC used for low quality frame as a function $R_W(l, l_F)$. In addition, the streaming bitrate is limited by the capacity constraint, T. So the model can be used to optimize the weighted playable frame rate, R_W, using the equation:

$$\begin{cases} Maximize: \\ \quad R_W = R_W(v_Q, v_{QF}) \\ Subject\ to: \\ \quad G \cdot ((S_I(v_Q) + S_{IF}(v_{QF})) + N_P \cdot (S_P(v_Q) + S_{PF}(v_{QF})) \\ \qquad + N_B \cdot (S_B(v_Q) + S_{BF}(v_{QF}))) \leq T \\ \quad 1 \leq v_Q \leq 31 \\ \quad 1 \leq v_{QF} \leq 31 \end{cases} \tag{4.53}$$

Similar to previous algorithms, with fixed input values from network and MPEG video, the space of possible values for v_Q and v_{QF} can be exhaustively searched to determine the FEC quantization value and the scaling quantization value to yield the maximum playable frame rate under the capacity constraint.

4.6.6 Revisiting of Media Independent FEC with Quality Scaling (MIQS)

Unlike Media Dependent FEC, which adds redundant lower quality frames, Media Independent FEC adds redundant packets to packets of video frames and increase their successful transmission probabilities (see Section 4.4.3).

WPFR of I frames

Only one quality level I frame is sent to the receiver and there is no lower quality FEC frame. The FEC here is only used to increase the successfully frame transmission probability. If the receiver gets the I frame (maybe after FEC), it will play it, if not, it plays nothing. So, $H_I = q_I$, $L_I = 0$, and the weighted playable frame rate of I frame can be predicted with:

$$R_{W_I} = G * ((1 - D(v_Q)) * q_I \tag{4.54}$$

WPFR of P frames

For every P frame, P_i, can only be displayed with a high quality when its preceding I or P frame is playable and itself are successfully received. Otherwise, it plays nothing. So,

$$\begin{cases} H_{P_i} = H_{P_{i-1}} * q_P \\ L_{P_i} = 0 \end{cases} \tag{4.55}$$

and

$$R_{W_{P_i}} = R_I * {q_P}^i \tag{4.56}$$

WPFR of B frames

Similar to the I and P frames, the B frame can either be played with high quality or nothing. Thus:

$$\begin{cases} H_{B_{ij}} = H_{P_{i+1}} * q_B \\ L_{B_{ij}} = 0 \end{cases} \tag{4.57}$$

and

$$R_{W_{B_{ij}}} = R_{W_{P_{i+1}}} * q_B \tag{4.58}$$

Total Weighted Playable Frame Rate

The total weighted playable frame rate is the sum of the rates for each frame type:

$$R_W = R_{W_I} + R_{W_P} + R_{W_B} = R_{W_I} + \sum_i R_{W_{P_i}} + \sum_{i,j} R_{W_{B_{ij}}} \tag{4.59}$$

Notice, after taking the common factor $(1 - D(v_Q))$ out from each of the I, P and B components, R_W is equal to the R_D used in our previous MIQS model (Section 4.4) and MITQS model (Section 4.5).

$$R_D = (1 - D(v_Q)) * R \tag{4.60}$$

Optimal Weighted Playable Frame Rate

Similarly, for given network and video conditions, the total weighted playable frame rate R_W for MIQS varies with the quantization value and the amount of FEC as a function $R_W(v_Q, S_{IF}, S_{PF}, S_{BF})$. In addition, the streaming bitrate is limited by the capacity constraint. So the model can be used to optimize the weighted playable frame rate, R_W, using the equation:

$$
\begin{cases}
Maximize: \\
\quad R_W = R_W(v_Q, S_{I_F}, S_{P_F}, S_{B_F}) \\
Subject\ to: \\
\quad G \cdot ((S_I(v_Q) + S_{I_F}) + N_P \cdot (S_P(v_Q) + S_{P_F}) \\
\qquad + N_B \cdot (S_B(v_Q) + S_{B_F})) \leq T \\
\quad 1 \leq v_Q \leq 31
\end{cases}
\tag{4.61}
$$

With fixed input values from the network and MPEG video, the space of possible values for v_Q and S_{IF}, S_{PF}, S_{BF} can be exhaustively searched to determine the FEC amounts and the quantization value to yield the maximum playable frame rate under the capacity constraint.

4.6.7 Comparison of MDQS and MIQS

Methodology

Using the optimization algorithms of MDQS (Equation 4.53) and MIQS (Equation 4.61), the weighted playable frame rates over a range of network and MPEG settings are explored. For each set of network and MPEG parameters, the weighted playable frame rates are compared for MPEG streaming with Media Dependent FEC (MDFEC) and Media Independent FEC (MIFEC).

1. MDQS: Before transmission, the server employs the optimization algorithm based on the MDQS model to determine the MDFEC and scaling quantization values that produce the maximum weighted playable frame rate R_W and uses these for the entire video transmission.

2. MIQS: Before transmission, the server employs the optimization algorithm based on the MIQS model to determine the MIFEC amounts and scaling quantization values that produce the maximum weighted playable

frame rate R_W and uses these for the entire video transmission.

The total bitrate used by the MPEG video and FEC is scaled to meet a capacity constraint [61] using Quality Scaling.

System Settings

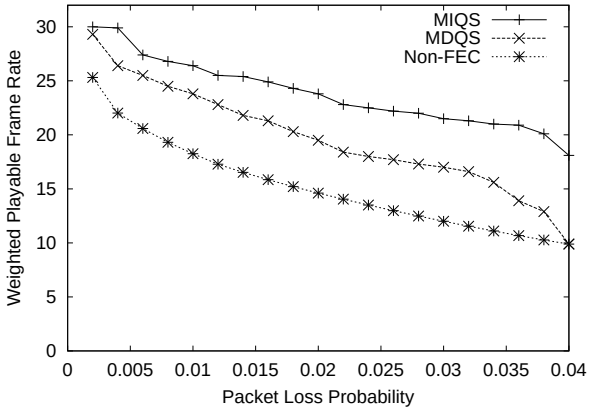
Network Layer		MPEG Layer	
t_{RTT}	50 ms	N_P	4 frames per GOP
s	1 Kbyte	N_B	10 frames per GOP
p	0.01 to 0.04	R_F	30 frames per second

Table 4.15: System Parameter Settings

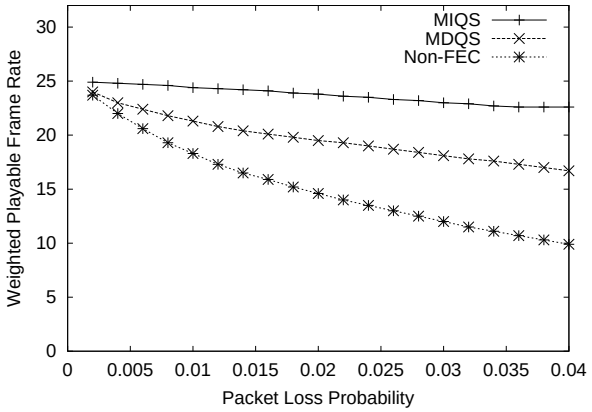
Table 4.15 presents the system parameter settings for the network and MPEG layers, similar to the MIQS study (Section 4.4). The video, *Paris*, is used and the relation between quality distortion and v_Q to a function as in Equation 4.26 on Page 4.26.

Results

The comparison of these two FEC methods with Quality Scaling is depicted by Figure 4.15. These two FEC methods are compared with the non-FEC method as well. The x-axes are the packet loss rates and the y-axes are the weighted playable frame rates. All the FEC and non-FEC methods are optimized with Quality Scaling under the TCP-Friendly capacity constraint in Figure 4.15a or 1.2 Mbps fixed capacity constraint in Figure 4.15b. Both of the figures show that both FEC methods yield better video quality, measured by weighted playable frame rate, than non-FEC for all loss rates. Moreover, this selection of loss rates allows us to show that the video quality of Media-Independent FEC, measured by weighted playable frame rate, is better than Media-Dependent FEC. The benefits of MIQS over MDQS are substantial, with MIQS providing 5-15 more frames per second for all rates.



a. TCP-Friendly Capacity Constraint



b. 1.2 Mbps fixed Capacity Constraint

Figure 4.15: Comparison of MDQS and MIQS

4.6.8 Summary

In this section, Media Dependent FEC with Quality Scaling (MDQS) for streaming MPEG is studied. A new video quality measure, weighted playable frame rate, is introduced to get the quality estimation. With this measure, a new optimization algorithm is built to adjust quality scaling levels and FEC patterns to maximize the video quality under the capacity constraint. Experiments show adjusted Media Dependent FEC with Quality Scaling has better quality than MPEG video without FEC.

Media Dependent FEC with Quality Scaling is compared to Media Independent FEC with Quality Scaling. Analytical experiments show that the video

quality, measured by weighted playable frame rate, of Media-Independent FEC is significant better than Media-Dependent FEC, with MIQS providing 5-15 more frames per second than MDQS over a range of network conditions. So we leave any additional study of Media-Dependent FEC as future work and focus on Media-Independent FEC.

Chapter 5

User Study

In Chapter 4, the ARMOR models were derived to analytically estimate the quality of streaming video and the algorithms were built to provide the optimal FEC and scaling pattern for video streaming. In this chapter, the ARMOR quality models, distorted playable frame rate R_D and playable frame rate R, are evaluated with a user study. Individual users were asked to carefully view selected video clips, which are representative of various streaming videos with FEC and scaling patterns chosen using our model. The users are also asked to view the original clips, without degradation before network transmission and rate the quality difference.

As illustrated in Table 4.1 on Page 50, an ARMOR system[1] can be divided into three layers: MPEG layer, ARMOR layer and network layer. Typically, the system obtains the parameters from the MPEG and network layers, then uses this information to adjust the variables in the ARMOR layer to optimize the streaming quality. In relation to these layers, the user study results are used to address several key research questions:

From the perspective of ARMOR,

[1] We define "ARMOR system" as a streaming system which uses ARMOR to adjust FEC and media scaling.

1. Is the distorted playable frame rate R_D an accurate video quality metric for capturing temporal and quality distortion, comparing to other metrics such as VQM and PSNR?

2. Is the playable frame rate R able to accurately capture temporal distortion?

3. Is Adjusted FEC an effective method for increasing perceptual quality of streaming video under a capacity constraint?

4. What are the differences in perceptual quality for Temporal Scaling and Quality Scaling?

From the perspective of MPEG and network layers,

5. Are the motion characteristics of a video correlated to streaming quality?

6. How critical is packet loss rate to streaming video quality?

5.1 Methodology

Our goal is to investigate how FEC and media scaling impact video distortion as compared to the original video. Thus, we use the double-stimulus impairment scale (DSIS) method in our tests with some modifications (see Section 2.5.5 on Page 33 for more details about the DSIS method). Specifically, an original clip is first presented with a frame rate of 30 frame per second, followed by the same clip degraded. After viewing the pair of clips, the user is asked to rate the second clip compared to the first on a five-point degradation scale from "Same" to "Much worse" without additional labels to avoid numeric biasing.

A small Visual C++ application was created as a test harness for the user study. The user can download the application bundled with the videos

from the author's Webpage space to any Windows PC and execute the test locally. The application first gathers user demographics and provides study directions. It then displays 17 pairs of 10-second video clips and asks the user to rate the differences in perceptual quality. The study takes about 8-10 minutes to complete, with the option to close the application in the middle of the experiment, if so desired. After the user exits the study, the user is asked to email the results to the author. All participants were eligible for a raffle of one $50, two $25 and five $10 BestBuy gift certificates and there was also extra credit given for the participants in one course.

5.2 User Study Application

When the application starts, a dialog box shows up to collect the user's demographics including gender, age, major, status, experience in viewing computer videos, and computer monitor type. Figure 5.1 shows a screenshot of the actual dialog box, where CS stands for "Computer Science" major, ECE stands for "Electrical and Computer Engineering" major, and IMGD stands for "Interactive Media and Game Development" major. A system call is also used in the background to retrieve each user's screen resolution.

The application then shows another dialog box (Figure 5.2) to provide directions to the user.

Next, the application opens the main dialog box which displays the video clips and asks the user to compare the quality of the degraded clip to the original clip. Figure 5.3 gives a screenshot of the main dialog box.

After the user completes the experiment, *Notepad* is opened with the results recorded and the user is asked to copy all the results and send them to the author via email.

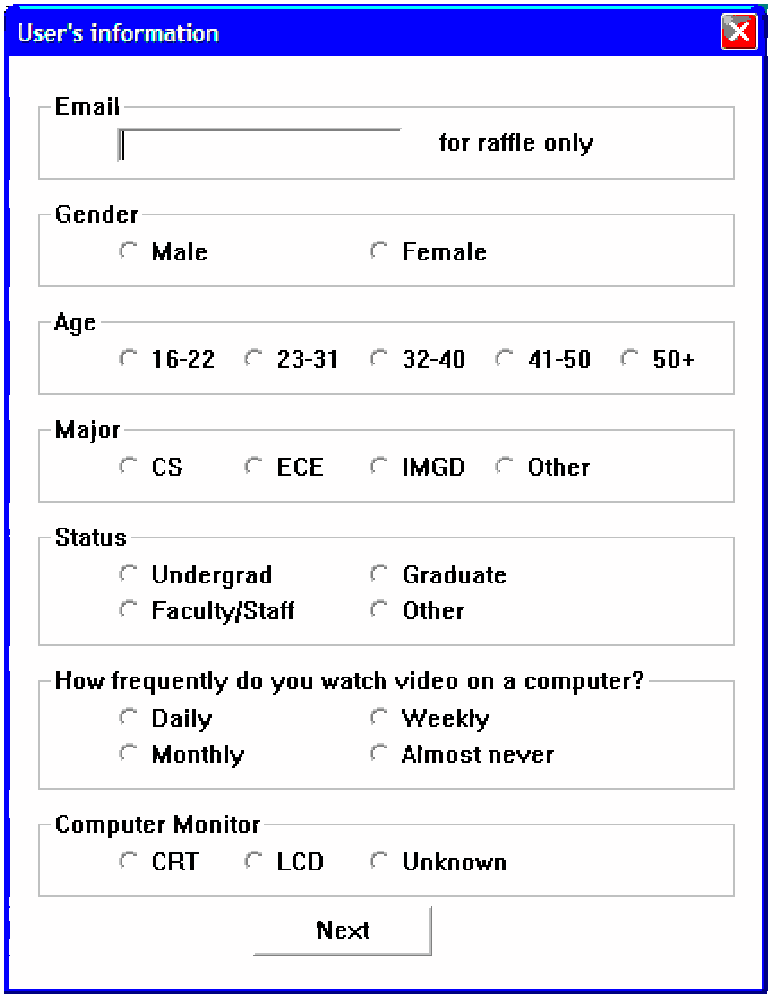


Figure 5.1: Screenshot of the User Information Dialog

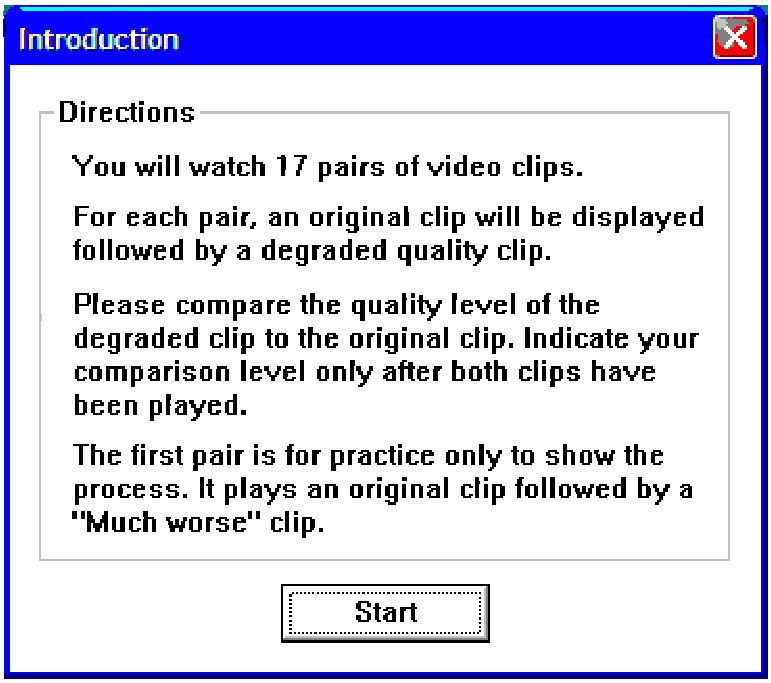


Figure 5.2: Screenshot of the Directions

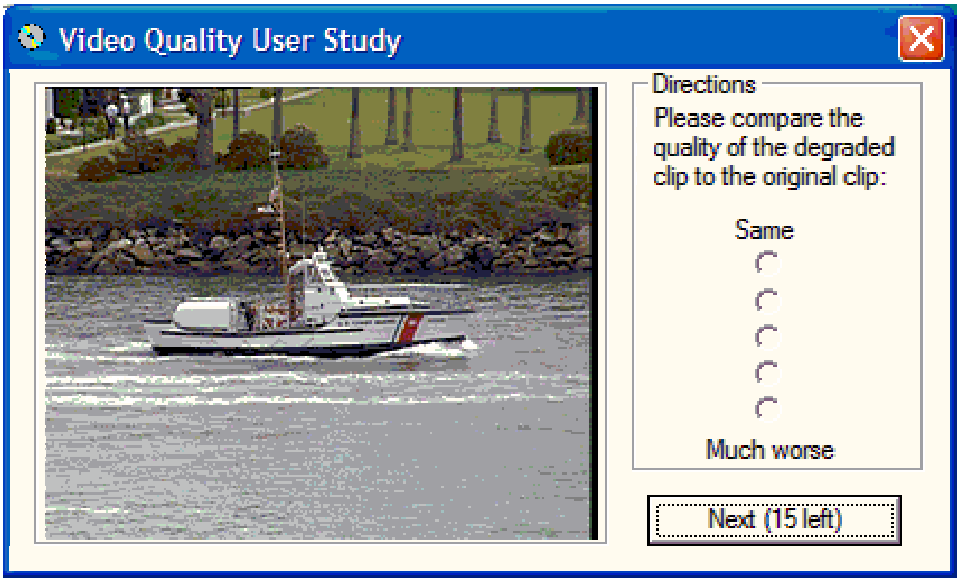


Figure 5.3: Screenshot of the Main Dialog, where users watch a pair of video clips and compare the second clip to the first clip.

5.3 Video Clips

A total of 17 pairs of video clips are used in this study. The first pair of video is used for training purposes so the user can get a better understanding of the evaluation system and to provide a baseline for the "Worst" quality tested. For this training, an original clip is played followed by the same clip severely degraded. The user is only able to choose the "much worse" option. The next 16 pairs are really evaluated by the users. For each clip pair, the first clip is the original with the best quality, and the second clip is degraded with all combinations of the following four independent factors:

1. Video Content: low motion clip *News (NW)* or high motion clip *Coast-guard(CG)* (see Table 4.2 on Page 65 for more details)
2. Packet loss rate: low loss rate *1%* or high loss rate *4%*
3. Repair: *Adjusted FEC (AFEC)* or *Non-FEC (NFEC)*
4. Scaling method: *Temporal Scaling (TS)* or *Quality Scaling (QS)*

where the first two factors come from the MPEG and network layer, and the second two factors come from the ARMOR layer.

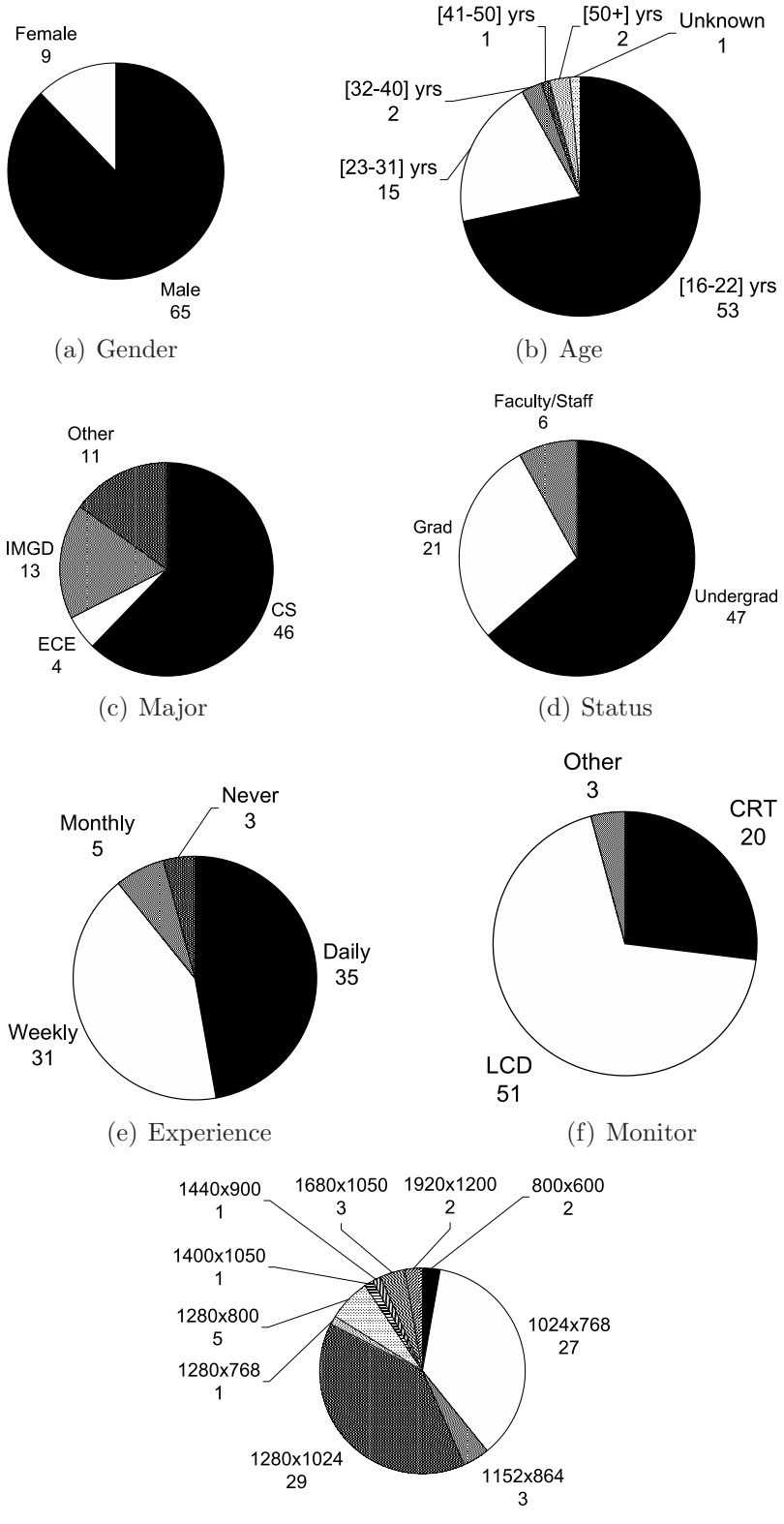
Table 5.1 shows the factor combinations of these 16 video clips. It also shows the playable frame rate R, quantization distortion D, and the distorted playable frame rate R_D derived from the models. For all the degraded clips, the total bandwidth used by the video and FEC is scaled to meet a TCP-Friendly constraint with Temporal Scaling or Quality Scaling, with 1.76 Mbps available for 1% loss and 0.69 Mbps available for 4% loss.

VID	MPEG and Network Layers		ARMOR Layer		R	D	R_D
	Video Content	Loss	Repair	Scaling			
v00	NW	1%	NFEC	TS	20.0	0.09	18.2
v01	NW	1%	NFEC	QS	23.7	0.13	20.5
v02	NW	1%	AFEC	TS	30.0	0.09	27.2
v03	NW	1%	AFEC	QS	30.0	0.09	27.2
v04	NW	4%	NFEC	TS	2.4	0.09	2.2
v05	NW	4%	NFEC	QS	20.1	0.37	12.6
v06	NW	4%	AFEC	TS	7.2	0.09	6.5
v07	NW	4%	AFEC	QS	29.6	0.25	22.3
v08	CG	1%	NFEC	TS	4.4	0.06	4.2
v09	CG	1%	NFEC	QS	22.9	0.27	16.8
v10	CG	1%	AFEC	TS	8.0	0.06	7.5
v11	CG	1%	AFEC	QS	30.0	0.27	21.9
v12	CG	4%	NFEC	TS	0.6	0.06	0.6
v13	CG	4%	NFEC	QS	17.8	0.41	10.5
v14	CG	4%	AFEC	TS	2.0	0.06	1.9
v15	CG	4%	AFEC	QS	28.3	0.41	16.7

Table 5.1: Clip factor combination

5.4 User Demographics

A total of 74 users took part in the study and all finished the experiment completely. Figure 5.4(a) shows most of the users are male. Figure 5.4(b) shows 53 users are in their late teens and early twenties, 15 users are between 23 and 31 and only a few users are over 31. Figure 5.4(c) shows most users (46) are computer science students and quite a few (13) are *Interactive Media and Game Development (IMGD)* majors. Figure 5.4(d) shows over 60% of the users are undergraduate students and about one-third of the users are graduate students. Figure 5.4(e) shows about half of the users claim they watch computer video everyday and about 90% of the users watch computer video at least once a week. Figure 5.4(f) shows that two-third of users have an LCD monitor. Figure 5.4(g) shows that more than one-third of the users have a resolution of 1280x1024 and another one-third have a resolution of 1024x768.



(a) Gender (b) Age

(c) Major (d) Status

(e) Experience (f) Monitor

(g) Screen Resolution

Figure 5.4: User Demographics

5.5 Results

This section analyzes the data reported by users to address the questions posed at the beginning of this chapter. This section first evaluates the accuracy of the ARMOR quality models. It then evaluates the impact of adjusting ARMOR variables on perceptual video quality. Lastly, it compares the impact of MPEG and network parameters on video quality.

5.5.1 Video Quality Measurements

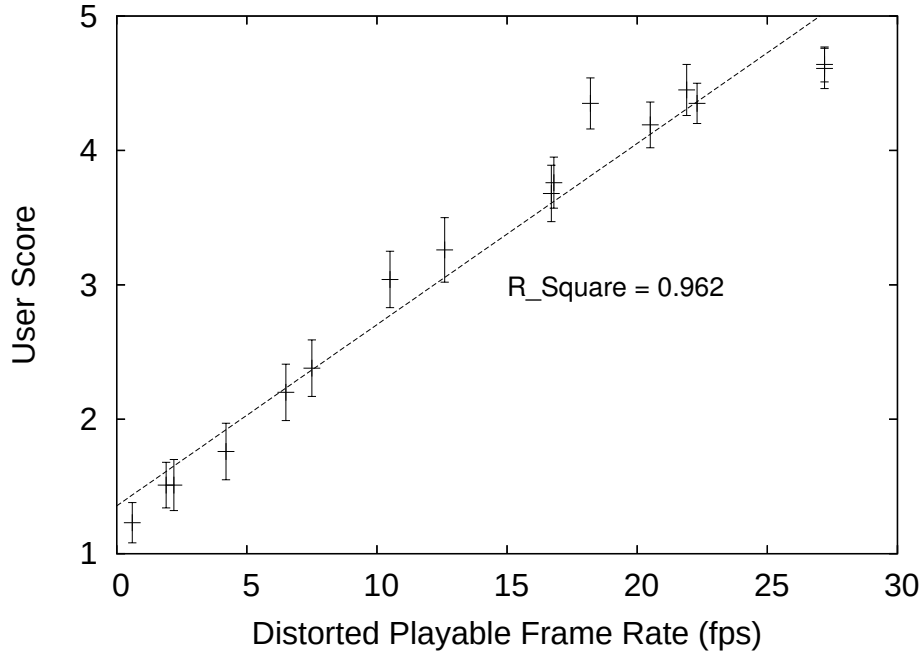


Figure 5.5: R_D vs. User Score

First, we analyze the data in regards to the first two questions: Is the distorted playable frame rate, R_D, an accurate video quality metric to capture temporal and quality distortion? Is the playable frame rate R able to accurately capture temporal distortion?

Figure 5.5 depicts the correlation of R_D and user perceptual quality. In the figure, each data point represents a video pair of the original clip and

the degraded clip. The x-axis is the distorted playable frame rate R_D derived from the ARMOR models and y-axis is the mean score achieved by all users from "Much worse"(1) to "Same"(5), shown with a 95% confidence interval. Visually, the relationship of R_D and user score is almost linear. When fitting a least squares error line with the data points, the R-Square value is 0.962. R-square means how much the line explains the amount of the total variation in the data and an R-Square value of 1 means a perfect line fit.

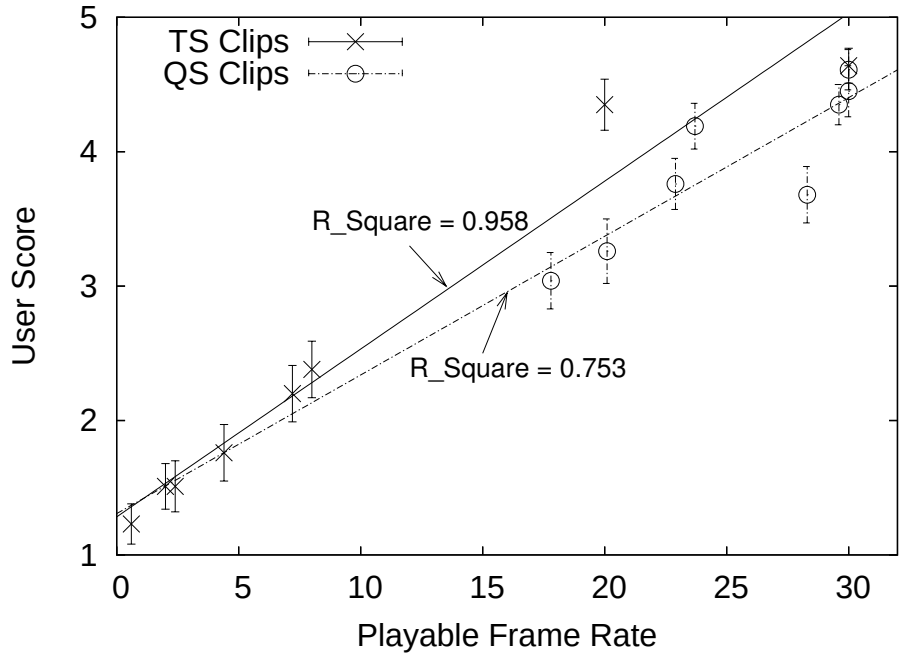


Figure 5.6: R vs. User Score

In the study of MITS, where Temporal Scaling is the only scaling method used, playable frame rate R is used as the video quality metric to estimate the user perceptual quality. Figure 5.6 depicts the correlation of R and user perceptual quality. In the figure, each data point represents a video pair of the original clip and the degraded clip, where each point with an 'x' mark means the clip is processed with Temporal Scaling and each point with an 'o' mark means the clip is processed with Quality Scaling. The x-axis is the playable

frame rate R derived from the ARMOR MITS model and y-axis is the mean score achieved by all users, shown with a 95% confidence interval. The solid line represents a least squares error line fit with the data points from temporal scaled clips, with a R-Square value of 0.958. The dashed line represents a least squares error line fit with the data points from quality scaled clips, with a R-Square value of 0.753. Visually, the correlation of R and user score for temporal scaled is almost linear and the correlation of R and user score is low for quality scaled clips.

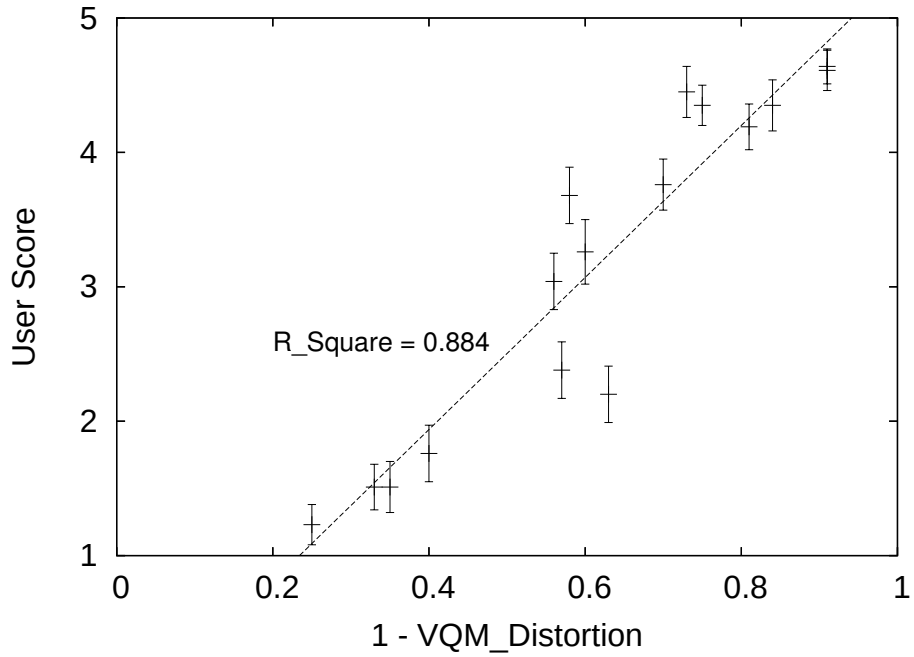


Figure 5.7: VQM Distortion vs. User Score

R_D and R is also compared with two other typical objective video quality metrics, VQM and PSNR, by showing how these metrics correlate to the user score. Figure 5.7 shows the VQM results. Similar to Figure 5.5, each data point represents a video pair and the y-axis is the mean user score with 95% confidence interval. Here, the x-axis is the distortion on the degraded clip measured by the VQM tool compared to the original clip. Visually, the

correlation between VQM and user score is not as strong. The figure also has a least squares line fit for the data points, and the R-Square value is 0.884.

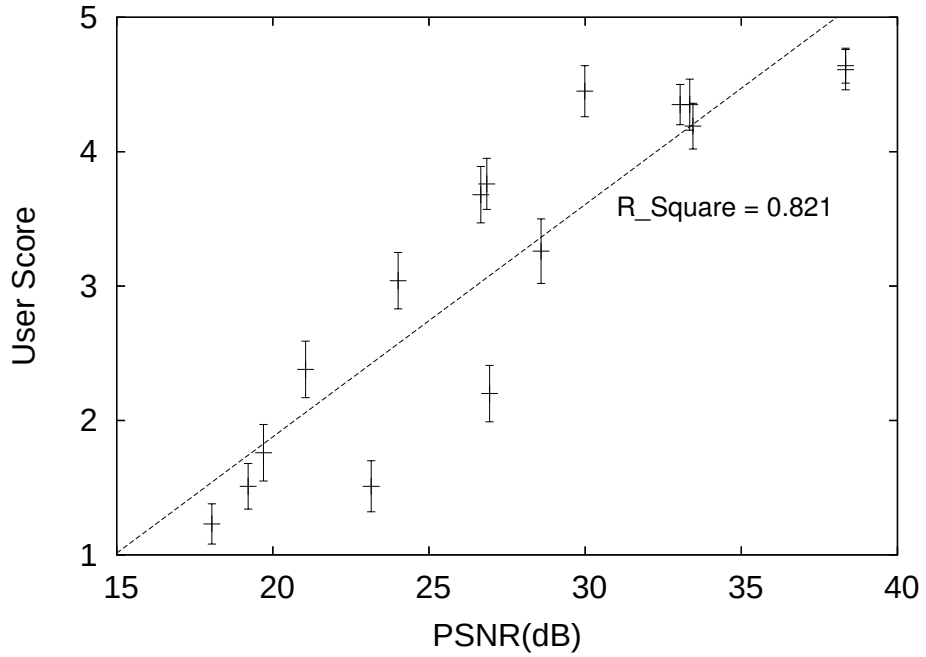


Figure 5.8: PSNR vs. User Score

Figure 5.8 shows the results for PSNR. Once again, each data point represents a video pair and the y-axis is the mean user score with 95% confidence interval. The x-axis is now PSNR computed with the original clip and the degraded clip. Visually, the correlation between PSNR and user score is even lower than that measured by VQM. The figure also has a least squares line fit for the data points, and the R-Square value is 0.821.

So the answers to the first two questions are, there is a high correlation between R_D and user perceptual quality, and R can effectively capture perceptual quality of the video with temporal distortion.

5.5.2 ARMOR Layer Variables

Given MPEG and network conditions, ARMOR can be used to optimize streaming video quality by adjusting the method and pattern of FEC and media scaling. In this section, the data is analyzed to evaluate the impact of FEC and media scaling on video quality. Specifically, we address the third question and the fourth question asked at the beginning of this chapter: Is Adjusted FEC an effective method to increase perceptual quality of streaming video under a capacity constraint? What are the differences in perceptual quality for Temporal Scaling and Quality Scaling?

Adjusted FEC

Figure 5.9 depicts the comparison of adjusted FEC and non-FEC. In this figure, the x-axis is the video clip instance and the y-axis is the mean score for all users shown 95% confidence intervals. The solid lines depict the clips with adjusted FEC and the dashed lines represent the clips without FEC. Clips with same x-axis value have the same values for the other three factors: video content, loss rate, and scaling method. The figure shows, in most cases, adjusted FEC yields a higher user perceptual quality than non-FEC no matter what the loss rate, video content and scaling method. There are two special cases, the first pair and the seventh pair, where the confidence intervals of adjusted FEC and non-FEC have overlaps. The first case is when the loss rate is low, TCP-Friendly capacity constraint is not strict, and the video has low motion. At that time, the quality without FEC is close to the best score and adjusted FEC can not significantly increase the quality. Another case is when the loss rate is high, capacity constraint is tight, and the video has high motion. At that time, the network and application conditions provide little bandwidth for adjusted FEC with Temporal Scaling. However, paired t-tests

conducted with SPSS[2] using the raw user data for both of these two cases show adjusted FEC provides a higher quality than non-FEC ($p < 0.003$).

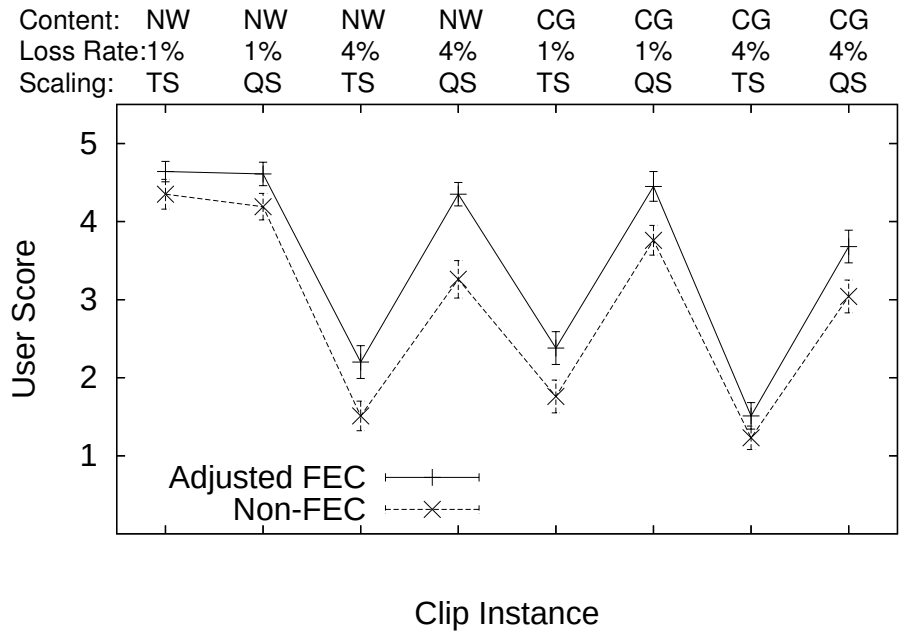


Figure 5.9: Adjusted FEC vs. Non-FEC

So the answer to the third question is, adjusted FEC can significantly increases perceptual quality on a network with packet loss.

Scaling Method

Figure 5.10 shows the comparison of Quality Scaling and Temporal Scaling. Similar to Figure 5.9, the x-axis is the video clip instance and the y-axis is the mean score for all users, shown with 95% confidence intervals. The solid lines depict the clips with Quality Scaling and the dashed lines represent the clips with Temporal Scaling. Clips with same x-axis value have the same values for the other three factors: video content, loss rate, and FEC method. The figure shows that Quality Scaling provides much better perceptual quality

[2]www.spss.com

than Temporal Scaling in most cases. Temporal Scaling is also beneficial to perceptual quality when the loss rate is low, the capacity constraint is relatively unconstrained, and the video does not have much motion.

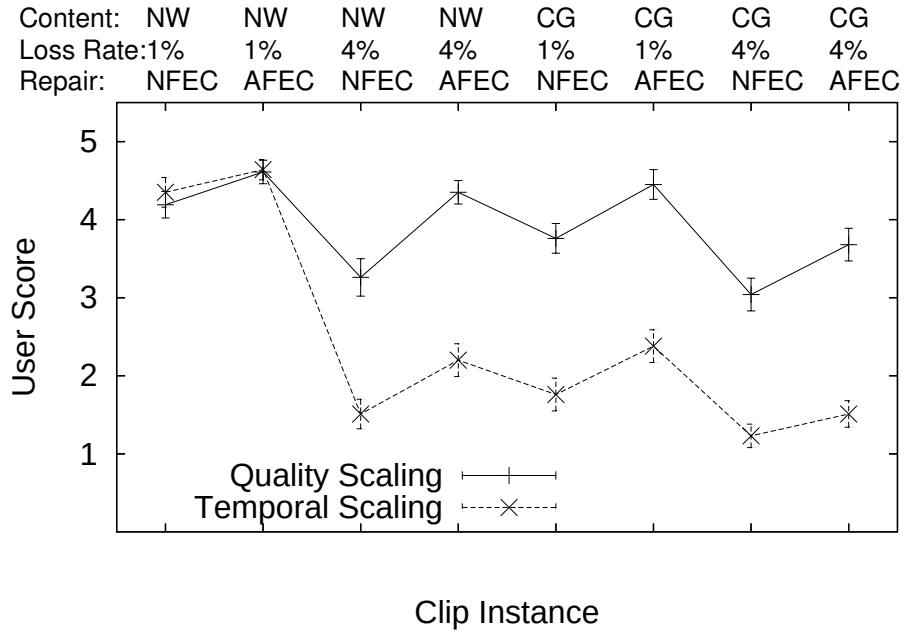


Figure 5.10: Temporal Scaling vs. Quality Scaling

So the answer to the fourth question is, the perceptual quality of video with Quality Scaling is relatively higher than Temporal Scaling.

5.5.3 MPEG and Network Parameters

Although ARMOR can be used to optimize streaming video quality, the quality is still highly correlated with MPEG and the underlying network conditions. In this section, the data is analyzed to compare the impact of MPEG and network conditions. Specifically, we address the fifth question and the sixth question asked at the beginning of this chapter: Are the motion characteristics of a video correlated to streaming quality? How critical is packet loss rate to streaming video quality?

Video Motion

Figure 5.11 shows the comparison of a high motion video *Coastguard* and a low motion video *News*. Similar to previous figures, the x-axis is the video clip instance and the y-axis is the mean score for all users, shown with 95% confidence intervals. The solid lines depict the *News* clips and the dashed lines represent the *Coastguard* clips. Clips with same x-axis value have the same values for the other three factors: loss rate, FEC method, and scaling method. The figure shows, low motion clips always have a better perceptual quality than high motion clips. Especially when loss rates are low and Temporal Scaling is used, the high motion clips have to discard more frames than low motion clips to satisfy the capacity constraints.

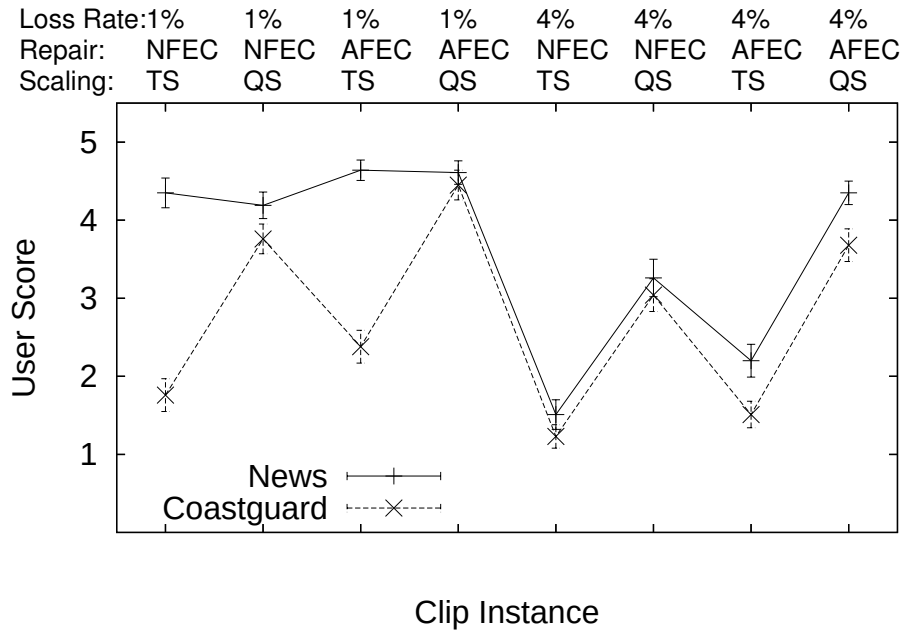


Figure 5.11: Low Motion Video *News* vs. High Motion Video *Coastguard*

So the answer to the fifth question is, the motion characteristics of a video are correlated to streaming quality and low motion clips have a better perceptual quality than high motion clips under bandwidth constraints.

Packet Loss

Figure 5.12 compares the clips streamed with 1% loss and 4% loss. Similar to previous figures, the x-axis is the video clip instance and the y-axis is the mean score for all users, shown with 95% confidence intervals. The solid lines depict the clips with 1% packet loss and the dashed lines represent the clips with 4% packet loss. Clips with same x-axis value have the same values for the other three factors: video content, FEC method, and scaling method. The figure shows that low loss rate always yields a better perceptual quality than does high loss rate, but the difference is not always significant. When Temporal Scaling is used with the low motion clips, *News*, the perceptual quality with low loss is acceptable and the perceptual quality with high loss is not acceptable. For high motion clips, *Coastguard*, the perceptual quality is similar for the clips with 1% loss and 4% loss since Quality Scaling and Adjusted FEC can effectively protect the video from loss.

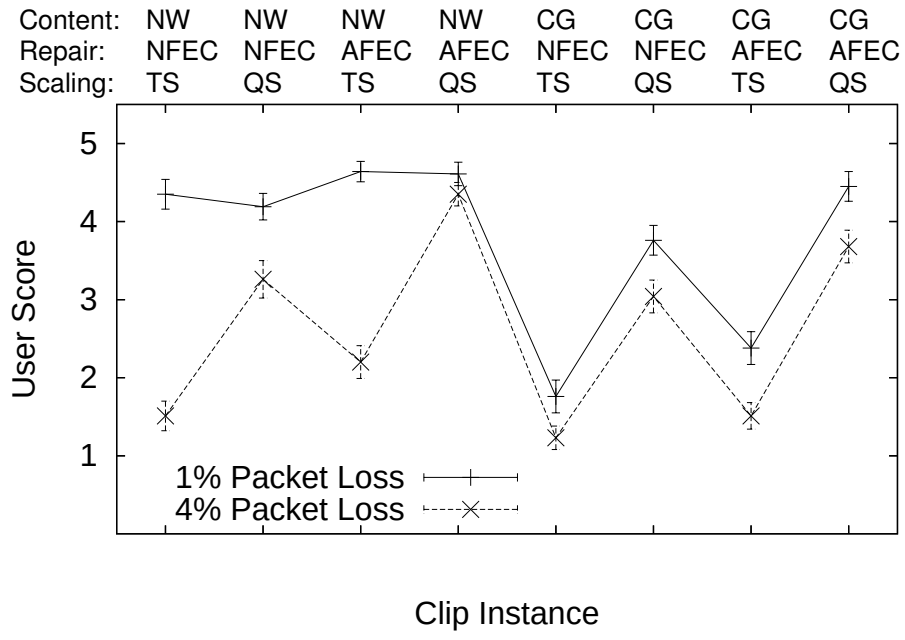


Figure 5.12: 1% Loss vs. 4% Loss

So the answer to the sixth question is, packet loss rate is essential, while not critical, to streaming video quality. Quality Scaling and Adjusted FEC can be used to effectively protect the video from loss.

5.6 Summary

This section presents analysis from a user study designed to measure video streaming quality with different MPEG and network conditions, and ARMOR choices of FEC and media scaling. A total of 74 users participated in the experiments and each of them compared the quality of 16 pairs of video clips.

The results illustrate:

1. The distorted playable frame rate R_D can be used to accurately reflect the user's perceptual quality. The correlation of R_D and user score is nearly linear and the R-Square value from a least square line fit is close to 1. The results also show that VQM and PSNR are not as accurate as R_D for estimating user perceptual quality and their R-Square from least square line fits are much lower than that of R_D.

2. The playable frame rate R is accurate in estimating the perceptual quality of temporal scaled videos, but is not accurate for quality scaled videos.

Second, the data are analyzed to evaluate the impact of ARMOR FEC and media scaling choices on streaming video quality. The results are:

3. Adjusted FEC can effectively improve the streaming quality for all loss rates, scaling methods, and video content.

4. Quality Scaling yields better perceptual quality than Temporal Scaling except when the loss rate is low and the video has low motion. In that

case, Quality Scaling provides the same perceptual quality as Temporal Scaling.

Third, the data are analyzed to compare the impact of MPEG and network conditions on video quality.

5. Low motion videos have better quality than high motion videos especially when the loss rate is low and Temporal Scaling is used.

6. Loss rate can significantly degrade video quality but the impact is not significant since Quality Scaling and Adjusted FEC can protect the video from loss.

Chapter 6

Simulations

ARMOR can be used as the core of streaming protocols that adjust FEC and media scaling in response to real-world application and network conditions. For the experiments in Chapter 4, the MPEG layer and network layer parameters remained fixed for the duration of each video. This simplified environment allows us to clearly illustrate the effects of adjusted FEC compared to that of fixed FEC and non-FEC approaches. However, in practice, MPEG video frame sizes change over the course of a video, and they may even change in the middle of a GOP. Moreover, while maximum network packet sizes are often fixed for the life of a flow, round-trip times and loss rates change rapidly and packet losses are often bursty.

This chapter explores the accuracy of ARMOR in predicting and optimizing video quality through exploring two specific cases in detail: Media Independent FEC with POst-encoding Temporal Scaling (MIPOTS), and Media Independent FEC with Quality Scaling (MIQS). Simulation experiments are designed to characterize more realistic network and video conditions. The video quality is measured by playable frame rate for MIPOTS and by distorted playable frame rate for MIQS. Comparing performance predicted and

optimized by ARMOR against simulated performance provides a strong indication of the effectiveness of using ARMOR models and algorithms within a streaming protocol in real Internet situations. Specifically, the analytic experiments assume:

1. An accurate estimate of the packet loss probability from the network protocol. Section 6.1 considers the effects of error in the packet loss estimate on ARMOR's predictive quality.

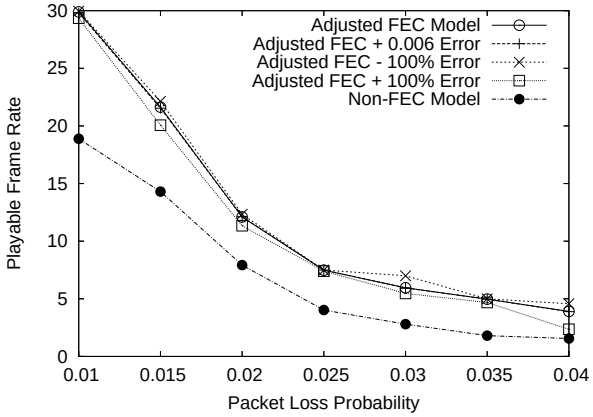
2. Independent network packet losses. Section 6.2 introduces bursty packet losses derived from previous Internet streaming measurements to determine the impact of the independent packet loss assumption on ARMOR's accuracy.

3. Fixed round-trip times for the life of the flow. Section 6.3 uses ARMOR to determine the appropriate media scaling assuming fixed round-trip times and then applies more realistic round-trip times obtained from traces of Internet streaming experiments.

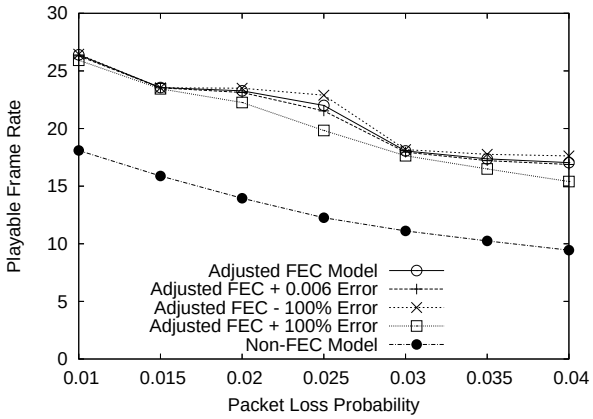
4. Constant I, P, and B frame sizes for the entire video. Section 6.4 uses ARMOR assuming a fixed frame size and then applies more realistic frame sizes based on traces from previous measurements of MPEG video.

For each experiment, the video quality predicted and optimized by ARMOR is compared to the actual quality achieved through the more realistic simulations. These comparisons evaluates the sensitivities of our models and algorithms are to real-world effects, while comparisons of video quality without FEC indicate the advantages of using ARMOR even if there are real-world inaccuracies. For all experiments, the values of system parameters held constant are the same as in the analytical experiments (see Table 4.5 for Temporal Scaling and Table 4.8 for Quality Scaling).

6.1 Inaccurate Loss Prediction



a. MIPOTS



b. MIQS

Figure 6.1: Impact of Inaccurate Loss Prediction

This simulation tests the effectiveness of using the adjusted FEC determined by the models and optimization algorithms when the loss rate is not accurately predicted. While under-predicting the loss rate results in too little FEC for effective repair, over-predicting the loss rate yields more FEC than necessary and leaves less available capacity for the MPEG data. Three sets of simulation experiments with different induced amounts of error in the loss probability prediction are run: 1) the actual loss rate is higher than the predicted loss rate by 0.006 which is the average margin for error found after

numerous simulations in [26]; 2) the actual loss rate is double the predicted loss rate; and 3) the actual loss rate is half the predicted loss rate.

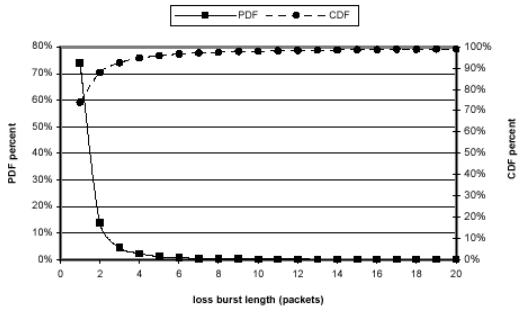
For each loss case, the predicted loss rate p is used in ARMOR to determine the FEC and scaling patterns. Then, we simulate streaming the MPEG video using these patterns on a network with the above actual loss rates and measure the actual playable frame rate at the receiver.

Figure 6.1 depicts the (distorted) playable frame rates for the simulations along with the playable frame rates estimated and optimized by ARMOR MIPOTS (Figure 6.1a) and ARMOR MIQS (Figure 6.1b). For the cases in which the actual error is underestimated, ARMOR's quality estimate does differ from the actual frame rate achieved, indicating that the inaccurate loss prediction does result in a slightly sub-optimal use of FEC. However, the actual frame rates achieved differ by less than 0.5 frame per second on average. Moreover, for the practical loss prediction errors of 0.006, the actual frame rates are nearly identical to the predicted frame rates. This suggests using ARMOR to determine proper FEC and scaling can be effective in practice.

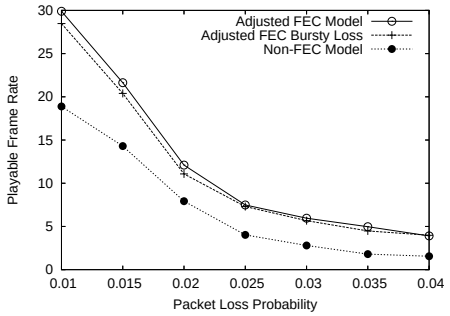
6.2 Bursty Loss

Our analytic models and algorithms assume independent packet loss events, while Internet packet losses are often bursty [48, 63]. Bursty losses may reduce the effectiveness of FEC especially when fewer than K of the N packets in a frame can be recovered and the resultant playable frame rate is lowered.

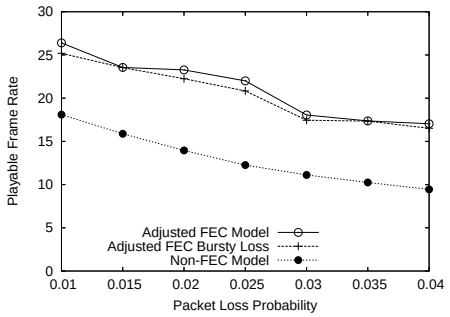
We use a series of traces from an Internet measurement study [15] to simulate the effects of bursty loss over a range of loss conditions. For each loss event, we use the probability distribution obtained from Internet streaming traces in [48] and given in Figure 6.2a to provide bursty loss events.



a. Loss Burst Distribution (from [48])



b. MIPOTS



c. MIQS

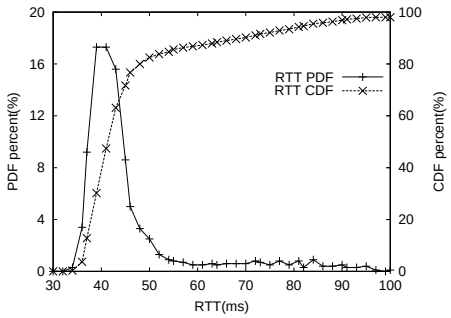
Figure 6.2: Impact of Bursty Loss

We use our models and optimization algorithms to determine the adjusted FEC and predicted (distorted) frame rate assuming independent losses. Then, we simulate streaming the MPEG video using the trace driven loss events and loss bursts and measure the actual playable frame rate at the receiver.

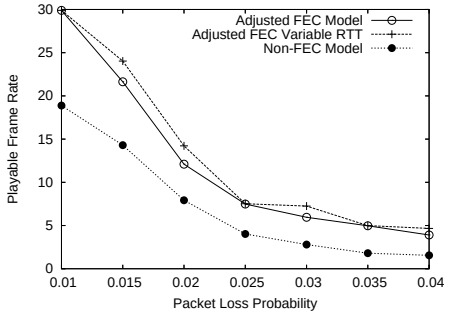
Figure 6.2 depicts the (distorted) playable frame rates for the simulations along with the playable frame rates estimated by ARMOR MIPOTS (b) and ARMOR MIQS (c). The bursty packet loss simulations both illustrate that the adjusted FEC models with independent loss assumptions marginally pre-

dict over-optimistic performance. However the differences are small enough to suggest that using ARMOR models and algorithms to determine adjusted FEC based on independent losses yields good performance in practice.

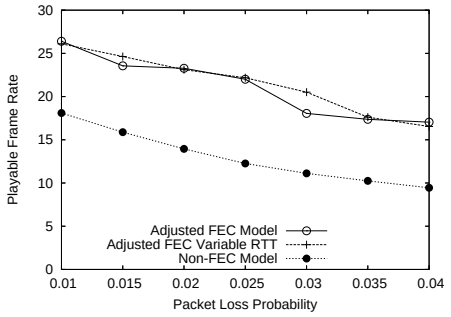
6.3 Variable Round-Trip Times



a. RTT Distribution (from [15])



b. MIPOTS



c. MIQS

Figure 6.3: Impact of Variable RTT

Our analytical models and search algorithms assume fixed round-trip times (RTTs) for the entire flow. In reality, RTTs can vary considerably during a

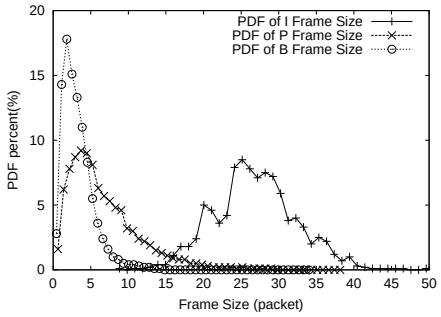
flow's lifetime. The possible impact of variable RTTs is that the bandwidth estimate using a fixed average RTT is inaccurate, and therefore this causes the choices for scaling and FEC to be less effective.

To study the effects of variable RTTs, we select a trace from [15], whose PDF and CDF are graphed in Figure 6.3a, that has an average RTT of about 45 milliseconds. We use ARMOR to determine the adjusted FEC and scaling patterns assuming a fixed RTT of 50 milliseconds. Then, we simulated streaming the MPEG video using the RTT trace and measured the actual frame playout rate at the receiver. To make the results comparable, each RTT from the trace is multiplied by 50/45 before the simulation so the average RTT of the simulation becomes 50 milliseconds.

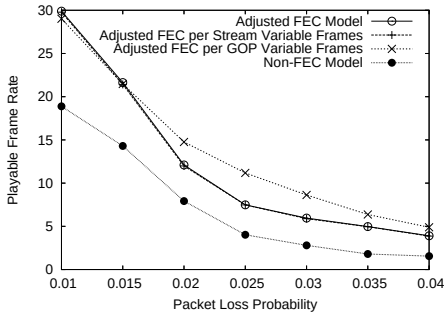
Figure 6.3b and Figure 6.3c provides the (distorted) playable frame rates for the simulations along with the (distorted) playable frame rates optimized by ARMOR MIPOTS and ARMOR MIQS. Surprisingly, the variable RTT curve has a slightly higher (distorted) playable frame rate than ARMOR estimated by using the average RTT. We attribute this to the fact that the RTT distribution selected is not Gaussian (normal), but instead has a somewhat heavy tail. Overall, even though the RTTs cover a wide range, the (distorted) playable frame rates estimated by ARMOR are close to the actual (distorted) playable frame rates, further suggesting ARMOR models and algorithms can be effective in practice.

6.4 Variable MPEG Frame Sizes

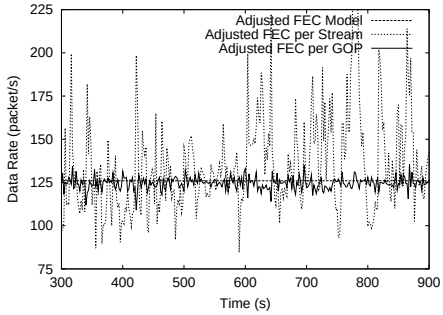
In the development of the analytic ARMOR models and algorithms, the MPEG frame size is assumed constant for the entire video. In reality, MPEG frame sizes change from frame to frame, even for the same type, and they may even



a. Frame Size Distributions (from [78])



b. MIPOTS



c. Data Rates for 2.5% Loss Rate

Figure 6.4: Impact of Variable Frame Size for MIPOTS

change inside one GOP. There are two possible impacts of variable-sized frames on the accuracy of ARMOR: 1) the adjusted FEC chosen using fixed average frame sizes will be inappropriate for the actual frame sizes and result in a lower playable frame rate; or 2) the FEC will have to be applied separately for each GOP, adding considerable processing overhead to the streaming application. To simulate the effects of variable MPEG frame sizes for ARMOR MIPOTS, we selected a frame size trace from [78]. Figure 6.4a presents the PDF distributions for frame types from this trace.

Once again, ARMOR was used to determine the adjusted FEC and scaling pattern assuming a fixed average frame size. Then, we simulated streaming the MPEG video with this FEC and scaling pattern, the frame sizes from the trace, and the network setting as in Table 4.5 on Page 80 to determine the actual playable frame rate at the receiver. Additionally, we applied ARMOR to each individual GOP, thus computing a new adjusted FEC based on the current GOP's I, P and B frame sizes. We simulated streaming the MPEG video using this per GOP adjusted FEC and measured the playable frame rate at the receiver.

Figure 6.4b graphs the playable frame rates for the simulations along with the playable frame rates estimated by our ARMOR MIPOTS. The frame rate depicted by adjusted FEC is almost the same as the adjusted FEC per stream simulation. At 2.0% loss rate and above, the simulation of adjusted FEC per GOP simulation produces a higher playable frame rate than all of the curves in Figure 6.4b.

Figure 6.4c focuses on the specific case of 2.5% loss to compare the simulated FEC scheme against data rates produced by ARMOR. Since ARMOR uses a fixed frame size, it yields a constant data rate equal to the TCP-Friendly rate of 126 packets per second. While remaining TCP-friendly over long time periods, the adjusted FEC per stream simulation produces considerable variation in its data rate. The adjusted FEC per GOP simulation, however, has a much smoother data rate that is significantly closer to the predicted constant data rate. Note, smooth data rates are much easier for networks to manage than bursty data rates.

Similarly, Figure 6.5 graphs the distorted playable frame rates for the simulations along with the distorted playable frame rates estimated by the ARMOR MIQS. The frame rates depicted by adjusted FEC are almost the same as the

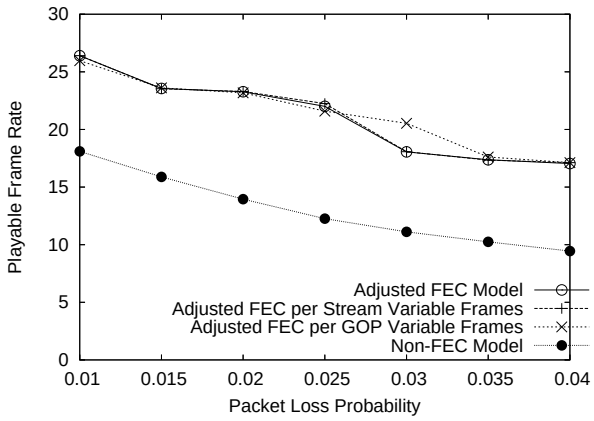


Figure 6.5: Impact of Variable Frame Size for MIQS

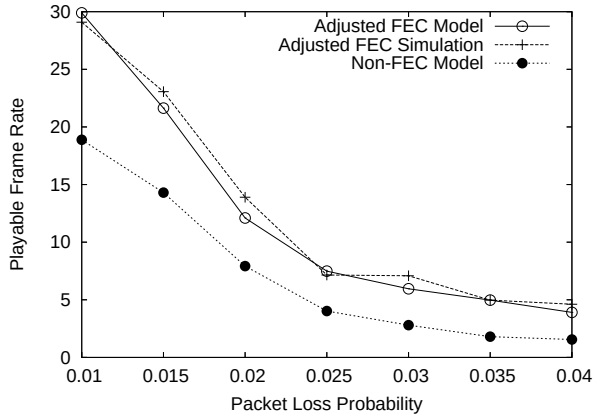
adjusted FEC per stream simulation. At 3.0% loss rate, the simulation of adjusted FEC per GOP simulation produces a higher playable frame rate than all the other curves in Figure 6.5.

Based on the observation from Figure 6.4 and Figure 6.5, where adjusted FEC per GOP simulation has higher playable frame rate and smoother data rate, we suggest that ARMOR should be applied to every GOP. Since one instance of ARMOR model and algorithm calculation can be executed in much less time than the real-time playout time for a GOP (Section 4.1.4), this repetitive use of ARMOR is feasible.

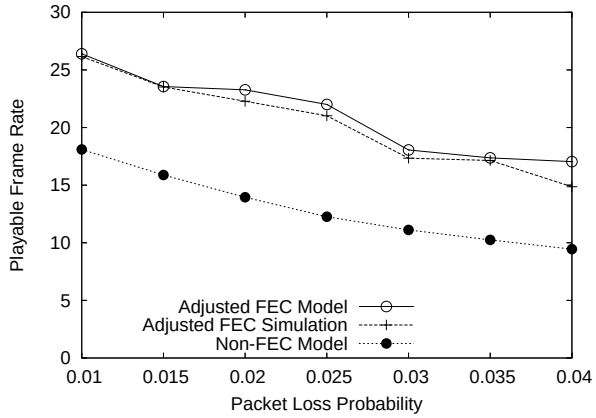
6.5 Combination Effects

The previous subsections show ARMOR to be resilient to mis-predictions for loss rate, RTT, or frame sizes. In this section, we simulated the effect of the combination of all three of these mis-predictions on ARMOR. Specifically, simulation loss rate is set higher than the predicted loss rate by 0.6%, the simulation RTT is from the RTT trace in Section 6.3, and the simulation actual sizes come from the frame size trace in Section 6.4. The results from

this simulation are compared to the analytic ARMOR model and algorithm performance where ARMOR assumes a fixed loss rate, a fixed RTT, and fixed frame sizes.



a. MIPOTS



b. MIQS

Figure 6.6: Combination Effects

Figure 6.6a and b presents the (distorted) playable frame rates for the simulations along with the playable frame rates estimated by ARMOR. The combined effects are similar to the effect seen in previous sections where the (distorted) playable frame rates optimized by ARMOR are close to the simulated playable frame rates. This provides further evidence that ARMOR can be effective in practice.

6.6 Practical Considerations

If ARMOR is to be used for interactive video, there are three practical issues that need to be addressed: a) dynamically changing a GOP based on the network conditions; b) arbitrarily long GOPs that would make exhaustive search prohibitive; and c) the case when the network and MPEG parameters are not known ahead of time. We address each concern separately:

a) For interactive streaming media encoded on the fly, the encoder can change the GOP (e.g., vary the number of P and B frames in the GOP). However, our prior study (Section 4.1) demonstrates that with FEC and a typical GOP (such as IBBPBBPBBPBBPBB), adjusting the GOP does not improve the playable frame rate.

b) In theory, GOP's can be arbitrarily long. However, our practical GOP study (Section 4.2) explains how, in practice, the GOP length can be effectively bounded. These previous results consistently suggest two guidelines: 1) the number of B frames between two reference frames should be only one or two (except when limited further by the encoding and time constraints); and 2) there is little performance gain in having more than five P frames per GOP.

c) If the network and MPEG parameters are known in advance, a system can use ARMOR to pre-compute the optimal FEC and scaling pattern for some typical network conditions. If the network and MPEG parameters are not known in advance (such as for an interactive videoconference), the streaming application can keep weighted moving average estimators of the MPEG frame sizes, the packet loss rates, and round-trip times from the previous epoch as an estimate of the parameters to use for the next epoch. While these estimators are likely to introduce mis-predictions, the experiments in this chapter indicate that using the analytic ARMOR models and optimization algorithms with estimated (and therefore somewhat inaccurate) parameters still yield (distorted)

playable frame rates that are within 1.8 frame per second of ARMOR estimates where all parameters are known in advance. Moreover, the concept of locality of frame sizes in relation to adjacent GOP's in an epoch [31] would lead one to believe that an estimator based on a weighted moving average would be reasonably accurate. In Chapter 7, we build a real ARMOR streaming system using ARMOR-MIPOTS model and algorithm with estimated parameters from previous GOPs. The measurements over the system show the ARMOR system provides quality close to that estimated by the analytical experiments.

Chapter 7

Implementation

In this chapter, we describe an implementation of a video streaming system that uses ARMOR with Media-Independent FEC and POst-encoding Temporal Scaling on a real network to validate earlier simulations and evaluate ARMOR under realistic system and network conditions. We define "ARMOR system" as the streaming system that uses ARMOR as a control module to optimize the video quality by adjusting the scaling level and FEC amount.

The cumulative effect of this implementation and previous simulation (Chapter 6) lends credibility to the fact that using the ARMOR model to predict video quality and use the optimization algorithm to adjust FEC with media scaling can be effectively used to provide high quality streaming video.

7.1 Background

This ARMOR system extends the work of a three-student Major Qualifying Project (MQP) [9]. Their system consists of five major modules working together: ARMOR module, MPEG video encoder/decoder, FEC encoder/decoder, TCP-Friendly UDP sender/receiver, and network QOS retriever. The MPEG encoder encodes a live video source or a file to produce an MPEG video stream.

The MPEG player on the receiving end plays frames from the stream. The ARMOR module uses the MIPOTS model and algorithm that considers network conditions to apply the optimal amount of FEC packets to each video frame. Additionally, the ARMOR module applies Temporal Scaling to the stream to remove the least important frames, thereby reducing the data rate when necessary. The network QOS retriever provides packet loss probability and delay time to the ARMOR module. The UDP sender and receiver transmit the video stream at a TCP-friendly rate.

However, the students' system does not work as expected. Their results show that adjusted FEC with POst-encoding Temporal Scaling is no better than 10% fixed FEC for all types of videos [9]. We looked at the MQP system and found two major shortcomings:

1. FEC encoder/decoder: the students did not notice the importance of using a packet-level FEC and they used bit-level FEC. Thus, their ARMOR module could not adjust the FEC amount added to each frame. Their Adjusted FEC usually has the same or lesser amount of FEC as the 10% fixed FEC.

2. Frame transmission order: because of the inter-frame dependencies, the MPEG encoder sends frames in a different order than displaying. For example, a GOP of `IBBPBBPBB` will be sent in the order of `IPBBPBBBB`. This has not been considered in the MQP system and the last two B frames of each GOP are discarded.

In rectifying these problems, we kept most of the MQP system, such as the architecture, MPEG encoder/decoder, and UDP sender/receiver, after some modification. We updated some essential parts that are not implemented appropriately such as the FEC encoder/decoder and the frame transmission

code. Approximately half the implementation of the MQP has been redone to make the updated system function correctly.

7.2 System Modules

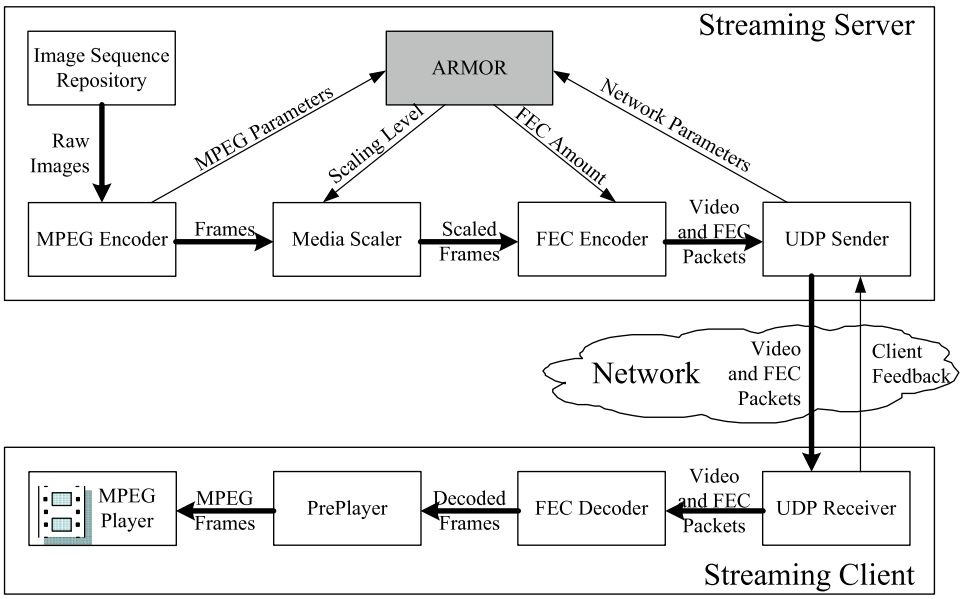


Figure 7.1: ARMOR System

The modules and information flows of the MIPOTS implementation are shown by Figure 7.1. There are two types of modules in the figure: 1) our ARMOR module, which is our MIPOTS model and optimization algorithm and is denoted by a gray box, and 2) all other data processing modules denoted by transparent boxes. The information flows can also be divided into two types: 1) data flows denoted by wide dark arrows, and 2) control flows denoted by thin arrows.

Considering the data modules and flows first:

1. A data flow starts from the image sequences repository on the server side. One sequence of raw images is encoded into video frames by an

FFMPEG[1] encoder. A simple ARMOR header is added before each MPEG frame, containing the frame size (in bytes), frame type (I, P or B), and frame sequence number.

2. The video frames, including their ARMOR headers, are then processed by a media scaler module, in this case doing POTS scaling, where some low priority frames might be discarded.

3. The FEC encoder takes the video frames, including their ARMOR headers, from the media scaler, splits each frame into packets, adds FEC packets for each video frame after the video data packets, and adds a packet header to each packet. For the FEC encoder/decoder, we used Luigi Rizzo's software FEC [77] and built a wrapper class over it to split the frame and to provide packet level FEC.

4. The UDP sender takes the video and FEC packets from the FEC encoder and sends them to the wide area network (shown by a cloud in Figure 7.1). The WAN is emulated by NistNet [13]. The UDP sender sends at a TCP-Friendly data rate determined by feedback on loss and RTT from the UDP receiver.

5. The UDP receiver at the client side receives video packets and FEC packets from NistNet with some packet losses. The loss rate and RTT are computed at the receiver in a sliding window over past five seconds. These parameters are reported to the UDP sender, then the ARMOR module every 200 milliseconds.

6. The FEC Decoder uses the video packets and FEC packets of each video frame to try to recover that frame. When the number of lost packets is

[1] http://ffmpeg.sourceforge.net/index.php

higher than the number of redundancy packets, the frame is dropped. Otherwise, it is playable and sent to the PrePlayer.

7. The PrePlayer module takes the decoded frames from FEC Decoder and removes the ARMOR headers. It also records some basic statistics such as: number of sent frames, number of received frames, and playable frame rates.

8. The FFMPEG player takes the playable MPEG frames from the PrePlayer and plays the video out on the screen.

The ARMOR module works in conjunction with these data processing modules. The ARMOR module takes the parameters of the video stream such as the GOP pattern and frame sizes, as well as the parameters from the transmission protocol and determines the best scaling level and FEC amount to produce the optimal quality. The ARMOR module then generates the optimal FEC and POTS combination and feeds this information to the video encoder and FEC encoder.

Figure 7.2 presents the data structure formats and use in detail. The system modules are denoted by round-corner boxes. The video data is denoted by white boxes and the ARMOR frame header and packet header are denoted by gray boxes. For example, the frame header has three fields: frame size, frame type and frame sequence number, and the packet header has four fields: packet size, the sequence number of the frame for each packet, the sequence number of this packet in the frame, and the FEC amount (N, K).

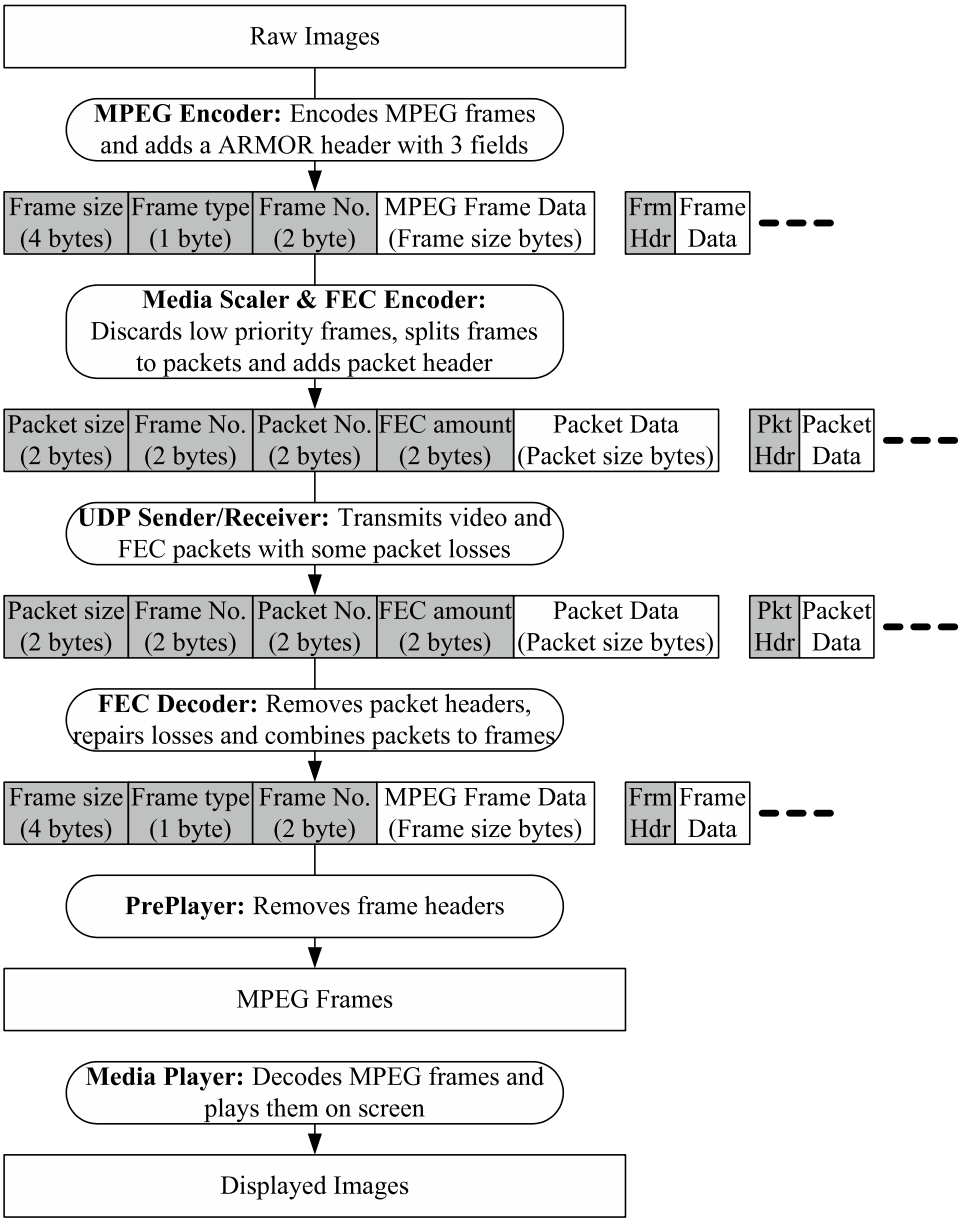


Figure 7.2: ARMOR System Data Formats

Network Layer		MPEG Layer			
t_{RTT}	50 ms	S_I	24.24 Kbytes (25 pkts)	N_P	3 frames per GOP
s	1 Kbyte	S_P	5.20 Kbytes (6 pkts)	N_B	8 frames per GOP
p	0.01 to 0.04	S_B	1.18 Kbytes (2 pkts)	R_F	30 frames per sec

Table 7.1: System Parameter Settings

7.3 System Testing and Evaluation

7.3.1 System Settings

Similar to the settings to MIPOTS for the analytical experiments (Section 4.1.4), Table 7.1 presents the system parameter settings for the network and MPEG layers. The packet size s, round-trip time t_{RTT} and packet loss probability p in the NistNet router are chosen based on the characteristics of many network connections [15, 38]. For all experiments, the parameters are fixed, except for packet loss probability p, which was varied from 0.01 to 0.04 in steps of 0.005.

A video clip, *Paris*, from PictureTel is used showing two people sitting at a table and talking while making high-motion gestures (see Figure 4.1 on Page 64). *Paris* has 1200 raw images and of size 352x288 pixels (CIF). A commonly-used MPEG GOP pattern, '`IBBPBBPBBPBBPBB`', and a typical full motion encoding frame rate R_F of 30 frames per second (fps) are used, providing an encoded MPEG video of 40 seconds. These settings yield average sizes of I, P and B frames of 24.24 KB, 5.20 KB and 1.18 KB, respectively. Rounding these frame sizes up to the nearest integer number of packets results in 25 packets, 6 packets and 2 packets for the I, P and B frames, respectively.

7.3.2 Loss Rate Prediction

For the experiments in Chapter 4, the network layer parameters remained fixed for the duration of each video. This simplified environment allowed us to clearly illustrate the effects of adjusted FEC compared to that of fixed FEC and

non-FEC approaches. However, in practice, these parameters change rapidly, especially the packet loss rate. This system assumes the network conditions are not known in advance, but must be detected by the UDP receiver and sent to the UDP sender periodically.

A probe packet is sent from the UDP sender to the receiver every ten milliseconds. The UDP receiver computes the loss rate by counting the number of lost probe packets in a sliding window over past five seconds and dividing it by the total number of sent probe packets in the same period, i.e. in the last five seconds. The length of 5 seconds for the average window was chosen after considering and trying the tradeoff between smoothness and responsiveness. The computed loss rate is reported to the UDP sender, and then the ARMOR module every 200 milliseconds. The round-trip time is estimated in a similar manner.

An alternative way to estimate the network parameters to use the video data packets without any probe packets. The receiver can count the past packet loss by checking the packet sequence numbers. With similar sampling and averaging methods, the receiver could report its estimated packet loss rate to the sender periodically. To get the round-trip time information, a timestamp would need to be added to each video or FEC packet, and the receiver should echo some, if not all, timestamps back to the sender allowing the sender to calculate the round-trip time easily.

Previous approach and other more accurate or efficient methods of estimating loss and round-trip time are beyond the scope of this thesis and are left as future work.

Figure 7.3 shows an example of loss rate prediction where the NistNet emulator's loss rate is set as 2%. The x-axis is time in seconds and the y-axis is the packet loss probability. The figure shows the predicted loss rate is close

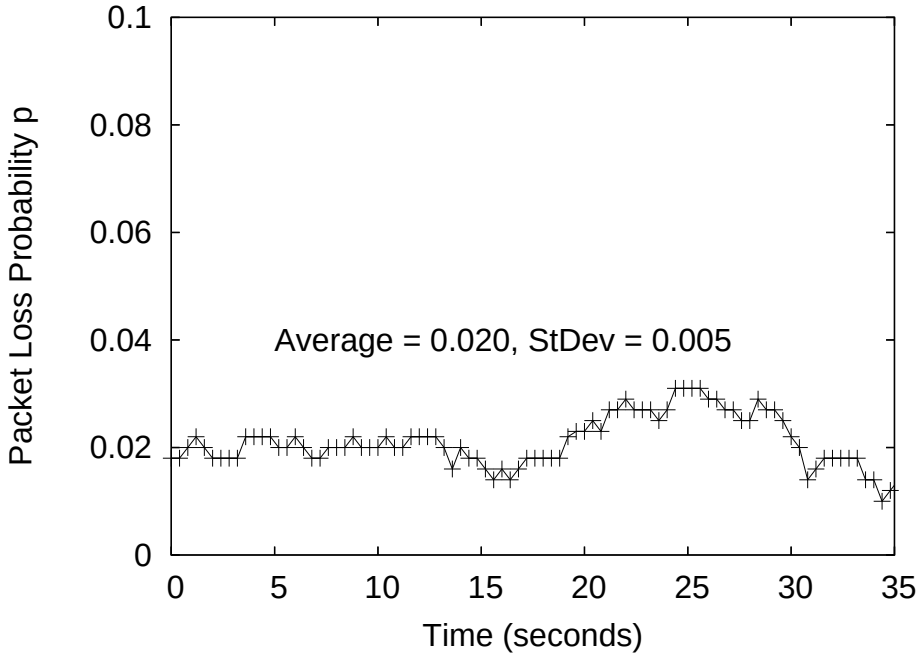


Figure 7.3: Loss Rate Prediction

to the actual packet loss value (0.02) with a small standard deviation (0.005). Notice, previous simulation experiments (Chapter 6) show the simulated frame rates are nearly identical to the predicted frame rates for a prediction error of 0.006.

7.3.3 TCP-Friendly Behavior

The ARMOR models, algorithms and systems are designed to optimize the quality of streaming video assuming a limited network capacity. This section verifies if the traffic of our implemented system is TCP-Friendly, and other characteristic performance as a network flow.

We ran experiments to compare ARMOR traffic to Wget[2], a publicly-available HTTP/FTP download application. The ARMOR system was streaming the *Paris* video. Wget was running at the ARMOR streaming client down-

[2]http://www.gnu.org/software/wget/wget.html

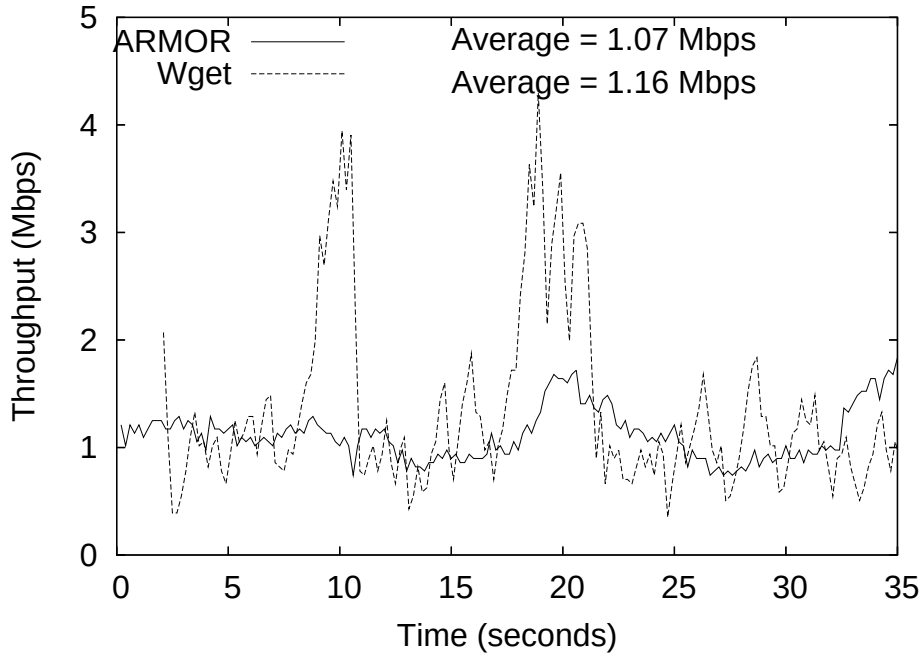


Figure 7.4: Throughput of ARMOR Traffic and Wget Traffic with 2% Loss

	ARMOR (Mbps)		Wget (Mbps)	
Run	Average	Std. Dev.	Average	Std. Dev.
1	1.07	0.17	1.16	0.57
2	1.09	0.19	1.14	0.64
3	1.02	0.16	1.12	0.46
4	1.00	0.24	1.12	0.68
5	1.15	0.23	1.38	0.75

Table 7.2: Throughput of ARMOR and Wget with 2% Loss for Five Runs

loading a 5.76 MB file from the ARMOR streaming server. Since the ARMOR system uses a packet size of 1 KB, the MTUs of the sender and receiver are set to 1 KB so Wget can use the same packet size. A Tcpdump[3] application runs at the client side to allow computation of throughput. The NistNet emulator's loss rate is set as 2%, the round-trip time is set to 50 ms, and the corresponding TCP-Friendly rate is 1.16 Mbps. The ARMOR traffic, Wget and Tcpdump applications are controlled to start and finish at approximately the same time.

[3]http://www.tcpdump.org/

Figure 7.4 shows one run of the experiments. The x-axis is the time with "0" denoting the starting time and the y-axis is the throughput of ARMOR traffic and Wget. Table 7.2 summarizes this run and provides statistics summary of four more runs. The figure and the table demonstrate the throughput of both of ARMOR and Wget are very close to the TCP-Friendly rate, with the ARMOR traffic being slightly more conservative than Wget. The data also shows that Wget has more variance than ARMOR since Wget uses TCP as a transmission protocol and adjusts its data rate every round-trip time while ARMOR adjusts its data rate every GOP, which is typically hundreds of milliseconds. This tradeoff between smoothness and responsiveness is inherent with every TCP-Friendly protocols [3, 26, 74] and further exploration of this tradeoff is left as future work.

7.3.4 System Performance Evaluation

In this section, we set up experiments over a range of NistNet loss rate settings to study the implemented ARMOR system's streaming quality. For each loss rate setting, the realistic playable frame rate measured at the receiver side is compared to the analytic ARMOR experiments and the ARMOR simulations. Details about these four schemes are as follows:

1. Analytical experiment with fixed loss rate and fixed frame sizes. This scheme uses the fixed loss rate of NistNet and the average frame sizes as in Table 7.1 for the whole streaming period and analytically calculates the optimal playable frame rate.

2. Simulation with fixed loss rate and estimated frame sizes. This scheme uses the fixed loss rate of NistNet for the whole streaming period. The real frame sizes encoded by the streaming server are read and the frame

sizes for the next GOP are predicted using a moving window average over the previous frame sizes. The playable frame rate for that GOP is optimized based on the fixed loss rate and estimated frame sizes. The average playable frame rate over all the GOPs is then computed.

3. Simulation with estimated loss rate and frame sizes. The packet loss rate are read from the UDP sender and the frame sizes are read from the encoder. The loss rate and frame sizes for next GOP are estimated using moving window average over previous information. The playable frame rate for that GOP is then analytically optimized with the estimated loss rate and frame sizes. Lastly, the average playable frame rate over all the GOPs is computed

4. Realistic ARMOR system measurement at the receiver side. This scheme optimizes the playable frame rates as in the third scheme. Moreover, for each GOP, the optimized scaling and FEC pattern is used to scale the video and add redundancy packets. The video is actually streamed over the emulated WAN (via NistNet) and the overall average playable frame rate is measured at the receiver side.

In all experiments, adjusted FEC is used to repair packet loss and POst-encoding (POTS) is used to satisfy TCP-friendly constraints.

Figure 7.5 depicts the playable frame rates for each of the four schemes. The x-axis is the packet loss probability, and the y-axis is the playable frame rate. The curves show the data for those four schemes from top to bottom. In the figure, the schemes are very close to each other, suggesting that ARMOR is robust in face of system inaccuracies. For some of the cases our analytical experiment's estimates do differ from the actual frame rates achieved by the real system, indicating that the inaccurate loss and frame sizes prediction does

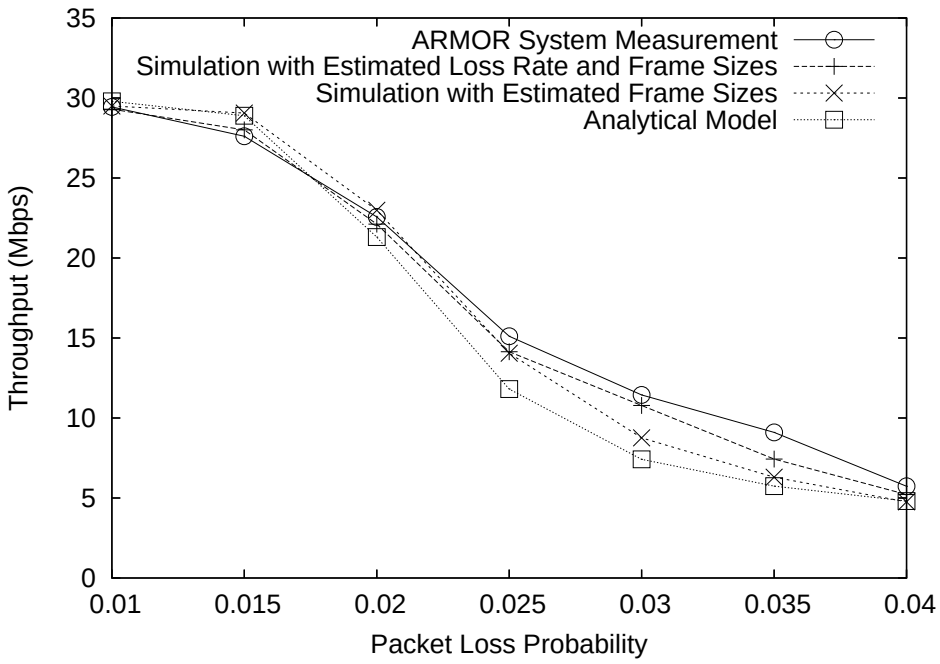


Figure 7.5: Performance of ARMOR System

result in a slightly sub-optimal use of FEC. However, by predicting the loss rate or/and frame sizes with the previous information for each GOP reduces the differences. The actual frame rates achieved differ by about 0.5 frames per second on average compared to the simulation with estimated loss rate and frame sizes. This suggests using our MIPOTS model and optimization algorithm to determine proper FEC and scaling can be effective in practice.

7.3.5 Comparison of FEC schemes

Using this implemented ARMOR system, the playable frame rates with different FEC methods over a range of network loss rates are explored. For each loss rate, the playable frame rates are compared for MPEG streaming without FEC, MPEG streaming with fixed FECs, and MPEG streaming with adjusted FEC. Similar to previous studies (Chapter 4), the following list gives the details about these four FEC choices:

1. Non-FEC: The sender adds no FEC to the video.

2. Fixed FEC (1/0/0): Each I frame receives 1 FEC packet. This simple FEC pattern protects the most important frame, the I frame.

3. Fixed FEC (3/2/1): The sender protects each I frame with 3 FEC packets, each P frame with 2 FEC packets and each B frame with 1 FEC packet[4].

4. Adjusted FEC: Before transmitting each GOP, the sender uses the ARMOR module to determine the FEC and temporal scaling patterns that produce the maximum playable frame rate and uses these for this GOP transmission

In all cases, the total bitrate used for each GOP is scaled to meet TCP-friendly constraints using POTS (Section 2.4.1).

Figure 7.6 graphs the playable frame rates for the four FEC choices for the *Paris* video. The x-axis is the packet loss probability, and the y-axis is the playable frame rate measured at the receiver side. From the data in this figure, adjusted FEC provides the best quality under all network and video conditions. The benefits of adjusted FEC over non-FEC are substantial, with adjusted FEC providing 5-10 more frames per second for all rates. The fixed FEC approaches usually improve playable frame rates over non-FEC video, and FEC(1/0/0) almost matches the playable frame rate provided by adjusted FEC for the 2.5% loss rate. However, the fixed FEC frame rates are still much lower than the frame rates with adjusted FEC for most cases. The two fixed FEC approaches overlap each other over this range of loss rate, similar to Figure 4.12b in Section 4.1.4 - MIPOTS. However, as discussed

[4] This FEC pattern provides strong protection to each frame and roughly represents the relative importance of the I, P and B frames. This adds approximately 15% overhead for each type of frame.

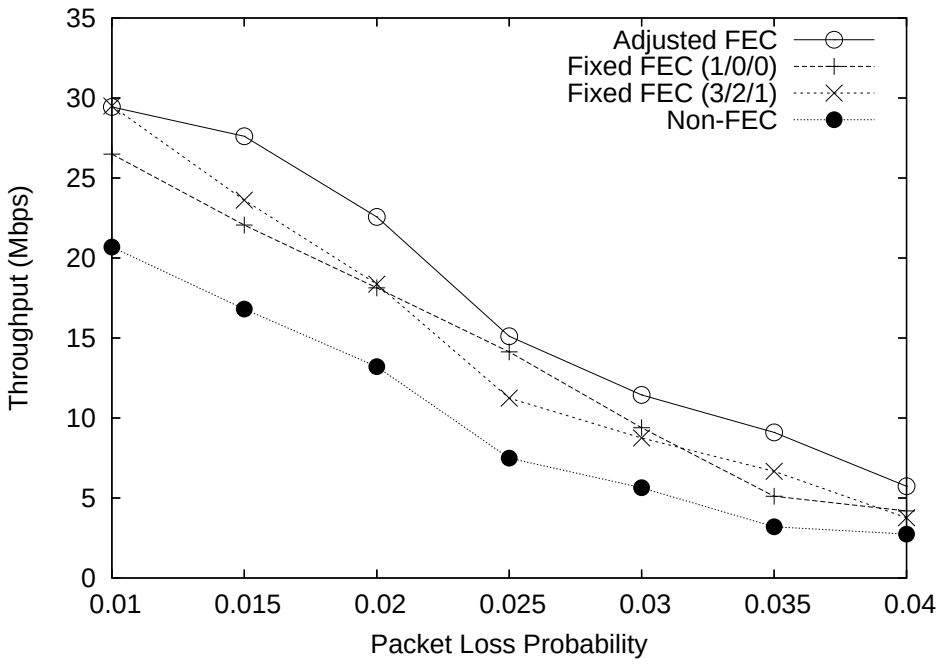


Figure 7.6: Comparison of FEC methods in Realistic ARMOR System

in Section 4.1.4, fixed FEC only works well for specific network and MPEG conditions, and both fixed FEC patterns are less effective than the more robust adjusted FEC when the network and application conditions change.

7.4 Summary

This chapter discusses the implementation of a working ARMOR system with our ARMOR module, FFMPEG encoder/decoder, Rizzo's FEC encoder/decoder, our FEC wrapper and TCP-Friendly UDP sender/receiver.

Using this system, the accuracy of loss rate prediction is studied and the traffic's TCP-Friendly behavior is measured. The results demonstrate that the system's loss rate prediction is close to the actual loss while the traffic is TCP-Friendly and smoother than TCP.

The system is them compared with the analytical experiment with fixed loss rate and frame sizes, the simulation with fixed loss rate and estimated

frame sizes, and the simulation with estimated loss rate and frame sizes. The results show the performances of these four schemes are close to each other. This suggests our MIPOTS model and optimization algorithm can be used effectively in practice to determine proper FEC and scaling pattern.

The adjusted FEC is also compared to other FEC methods with this ARMOR system implementation. The measurements at the receiver side indicate that adjusted FEC always achieves significantly higher quality than MPEG video without FEC and provides higher video quality than fixed FEC approaches when taken over a range of network conditions. This is consistent with previous analytical experiments in Chapter 4 and simulations in Chapter 6, and lends credibility to the fact that using the ARMOR model and algorithm to adjust FEC can effectively provide high quality streaming video.

Chapter 8

Conclusions

The volume of video data on the Internet is rapidly increasing and it raises the probability of streaming multimedia flows encountering bandwidth constraints and network packet loss. To match the capacity constraint while preserving real-time playout, media scaling can be used to discard the encoded multimedia content that has the least impact on perceived video quality. To limit the impact of lost packets, repair techniques, e.g. Forward Error Correction (FEC), can be used to repair frames damaged by packet loss. However, adding such data requires further reduction of multimedia data, making the decision of how much repair data to use of critical importance.

The *Adjusting Repair and Media Scaling with Operations Research (ARMOR)* models are designed to estimate the perceptual quality of streaming video with Forward Error Correction and media scaling, where FEC adds redundancy to recover packet loss and scaling reduces the streaming bitrate to satisfy capacity constraint. With the models, ARMOR algorithms are built to improve the perceptual quality of streaming video by optimizing the level of media scaling and the amount of FEC.

This thesis discusses the design and analysis of ARMOR models and algo-

rithms with mathematics derivation, simulation, system implementation and user studies.

8.1 Analytical Models and Optimization Algorithms

In Chapter 4, we analytically derived mathematical models to estimate perceptual quality for streaming video with Forward Error Correction (FEC) and Media Scaling in the presence of packet loss. With these models, operations research algorithms are built to determine the optimal adjustment of FEC, including Media Independent FEC (MIFEC) and Media Dependent FEC (MDFEC), and media scaling, including Temporal Scaling and Quality Scaling under capacity constraints.

8.1.1 Media Independent FEC with Temporal Scaling (MITS)

When Temporal Scaling is the only scaling method used and Media Independent FEC is used at the packet level, it can be assumed that each playable frame has the same quality since their encoding quantization values do not change and MIFEC can not recover partial frame. In this case, the playable frame rate, which represents how many video frames can be played at the receiver, can be used to measure the quality of the streamed video.

Initially, the successful transmission probability of each frame is computed based on the frame size, loss rate and the FEC amount. The dependencies among frame types are then used to determine if the received frames are playable. The total playable frame rate is added up as a prediction of

user perceptual quality. With this model, an optimization algorithm is built to adjust the amount of Media Independent FEC (MIFEC) and the level of Temporal Scaling, including POst-encoding Temporal Scaling (POTS) and Pre-Encoding Temporal Scaling (PETS), under a capacity constraint.

The analytic experiments employing the model and algorithm indicate that adjusting FEC with Temporal Scaling provides improvement over current approaches. When Temporal Scaling is used to to meet the TCP-friendly constraint, the adjusted FEC mechanism always achieves a higher playable frame rate than MPEG video without FEC and provides a higher playable frame rate than any fixed FEC approaches. The results also show, over a wide range of network and MPEG conditions, small fixed FEC and large fixed FEC are comparable.

8.1.2 Media Independent FEC with Quality Scaling (MIQS)

When a video is streamed over an unreliable network with Quality Scaling, its perceptual quality is degraded by two factors: quantization distortion and frame loss. Quantization distortion is caused by high quantization values and appears visually as coarse granularity in every frame. Frame loss is due to network packet loss and yields jerkiness in the video playout. A new quality measurement, distorted playable frame rate R_D, is presented to capture frame loss and quantization distortion. R_D uses the playable frame rate from the MITS quality model to estimate the frame loss and uses the Video Quality Metric (VQM) to estimate the quantization distortion. R_D then uses the product of these two values as the final quality measurement. A preliminary user study shows a high correlation between user perceptual quality and our distorted playable frame rate metric.

With this new measure of perceptual quality, an optimization algorithm

is built to adjust the amount of Media Independent FEC (MIFEC) and the level of Quality Scaling under a capacity constraint to maximize the distorted playable frame rate. The results show, when Quality Scaling is used to satisfy the TCP-Friendly capacity constraint, adjusted FEC has significant advantages over non-FEC and fixed FEC approaches when taken over a range of MPEG encoding and network conditions. The small fixed FEC approaches usually improve playable frame rates over non-FEC video but are still much less effective than adjusted FEC. Large FEC achieves the playable frame rate provided by adjusted FEC for low loss rates because the TCP-Friendly rate is relatively high, but with limited capacity (at high loss rates), the large FEC overhead becomes significant for Quality Scaling and none of the original video data is sent.

8.1.3 Media Independent FEC with Temporal and Quality Scaling (MITQS)

The distorted playable frame rate model can also be used to study the combination of both Temporal and Quality scaling with some adjustments. Specifically, the playable frame rate can be used to estimate the frame loss due to Temporal Scaling, packet loss and Forward Error Correction. The VQM distortion can be used to capture the quantization distortion from Quality Scaling. As in the previous cases, a new optimization algorithm is built to exhaustively search all possible combinations of scaling levels and FEC patterns to find the configuration that yields the best video quality under the capacity constraint.

Analytic experiments are conducted on nine videos, including *Paris*, with varied motion characteristics. The results show that when capacity constraints are moderate and loss rates are low, Temporal Scaling adds little to the quality produced by using only Quality Scaling. When bitrates are low and loss rates

are high, Temporal Scaling used to assist Quality Scaling provides improved video quality. Additionally, the results imply that Quality Scaling provides better quality video than Temporal Scaling and that differences in their performance is correlated to video motion characteristics. Under conditions with loss, adjusted FEC always achieves higher quality than MPEG video without FEC or any fixed FEC approach.

8.1.4 Media Dependent FEC with Quality Scaling (MDQS)

Since Media Dependent FEC can partially recover a lost frame, a new video quality measurement, weighted playable frame rate, is introduced as a more general replacement of distorted playable frame rate. Weighted playable frame rate models every frame separately and weights each frame based on its quantization distortion measured by VQM. Then the weights of all the frames are added up to get the quality estimation. With this measure, a new optimization algorithm is built to adjust quality scaling levels and FEC patterns to maximize the video quality under the capacity constraint. Experiments show adjusted Media Dependent FEC with Quality Scaling has better quality than MPEG video without FEC.

Media Independent FEC with Quality Scaling is revisited with this new quality measure, weighted playable frame rate, and compared to Media Dependent FEC with Quality Scaling. Analytical experiments show that the video quality, measured by weighted playable frame rate, of Media-Independent FEC is significant better than Media-Dependent FEC, with MIQS providing 5-15 more frames per second than MDQS over a range of network conditions.

8.2 User Study

A user study is designed to measure video streaming quality with different video content, scaling methods, repair methods and packet loss rates. A total of 74 users participated in the experiments and each of them evaluated the quality of 16 combinations of two video clips.

Analysis of the results shows the distorted playable frame rate can be used to accurately reflect the user's perceptual quality. The correlation of distorted playable frame rate and user score is close to linear with a small error. The results also show that VQM and PSNR are not as accurate a measure of user perceptual quality, having more error in least square line fits than the distorted playable frame rate.

The results also show: Adjusted FEC can effectively improve the streaming quality for all loss rates, scaling methods, and video content; Quality Scaling always yields better perceptual quality than Temporal Scaling except when the loss rate is low and the video has low motion; Low motion clips always have better quality than high motion clips especially when the loss rate is low and Temporal Scaling is used; Loss rate degrades video quality but the impact is not always significant since Quality Scaling and Adjusted FEC can protect the streaming video from loss.

8.3 Simulation

Simulation experiments are designed with MIPOTS and MIQS to test our models' and algorithms' accuracy in predicting and optimizing video quality with more realistic network and video conditions by:

1. Considering the effects of error in the packet loss estimate on our models' predictive quality.

2. Introducing bursty packet losses derived from previous Internet streaming measurements.

3. Applying more realistic round-trip times obtained from previous traces of Internet streaming measurements.

4. Applying more realistic frame sizes based on traces from previous measurements of MPEG video.

For each experiment, the video quality predicted by our analytic model and optimized by our search algorithm is compared to the actual quality achieved through the more realistic simulations.

All the results show our models and optimization algorithms are robust in the presence of real-world effects. The comparisons of video quality without FEC indicate the advantages of using our models and algorithms even if there are inaccuracies in the model brought on by the real-world.

8.4 Implementation

A realistic ARMOR MIPOTS system is implemented with our ARMOR model plus search algorithm, MPEG encoder/decoder, FEC encoder/decoder, and TCP-Friendly UDP sender/receiver.

With this realistic system, the accuracy of loss rate prediction and the traffic's TCP-Friendly behavior is studied. The results show that the system's loss rate prediction is reasonable and the ARMOR traffic is TCP-Friendly, while the bitrate consumed by ARMOR is more conservative and smooth than a bulk-transfer TCP flow.

The ARMOR system is compared with the analytical experiment for fixed loss rate and frame sizes, the simulation with fixed loss rate and estimated

frame sizes, and the simulation with estimated loss rate and frame sizes. The results show the performances of these four schemes are close to each other and demonstrate that our MIPOTS model and optimization algorithm can be used effectively in practice to determine proper FEC and scaling pattern.

The adjusted FEC method is compared to other FEC methods within this realistic ARMOR system. The measurements at the receiver side show that adjusted FEC always achieves significantly higher quality than MPEG video without FEC and provides higher video quality than fixed FEC approaches when taken over a range of network conditions. This lends credibility to the fact that using our ARMOR model and algorithm to adjusted FEC can practically improve the quality of streaming video. Similar to analytical MIPOTS experiment, over a range of network and MPEG conditions, small fixed FEC and large fixed FEC are comparable.

Chapter 9

Future Work

There are some areas of future work that can be extended from this research.

1. The MITS study uses playable frame rate as the quality measurement without considering the variance of frame rate. Possible future work could incorporate the impact of variance in frame rate into our model and algorithm. For example, if a low variance is more important than a high playable frame rate, only Temporal Scaling patterns that evenly distribute the discarded frames can be considered.

2. In the practical GOP study, a large value of δ makes the effective GOP length increase. With the extended GOP, when an I frame is lost, a long gap appears during playout since the following frames are not decodable until the next I frame arrives successfully. Dynamically sized GOPs could possibly be used to address this problem. For example, when δ is high, the GOP pattern could be reduced to I frames only so a lost I frame does not introduce lengthy propagation errors.

3. The MIQS and MITQS studies assume that I, P and B frames always have the same quantization values. Since it is possible to use different

values for different frame types, future work could seek to capture the quality dependencies among frames with different quantization values.

4. In the MIQS and MITQS studies, VQM distortion and frame sizes are approximated as exponential functions of the quantization value. However, analyzing videos and fitting the results to these exponential functions is a time-intensive operation. It may be possible to analyze a large variety of videos and find parameters for the exponential functions, based on the frames sizes, frame rates and video content, that are effective in general.

5. Analytical experiments used to compare MDQS to MIQS show MIQS outperforms MDQS for all situations. However it might be that the results are only valid for certain network and application conditions. An alternative evaluation approach could use mathematical derivation to prove this relationship analytically with the quality functions for MDQS and MIQS.

6. The implementation of the ARMOR MIPOTS system gives confidence that ARMOR can be used in the real-world and that the system performance is consistent to that shown in analytical experiments. Future work could implement the MIQS and MITQS systems. These new implemented systems should be able to lend more credibility to the fact that using our ARMOR model and algorithm to adjust FEC can practically improve the quality of streaming video. These systems could also be compared to each other in a practical environments.

Other future work could include a study of more accurate and efficient network estimations, user studies with more videos or higher quality videos, a study of audio streaming, a study of FEC with retransmission, and a study of the tradeoff between smoothness and responsiveness of ARMOR flow.

Bibliography

[1] A. Albanese, J. Bomer, J. Edmonds, M. Luby, and M. Sudan. Priority Encoding Transmission. *IEEE Transactions on Information Theory*, 42(6):1737 – 1744, Nov. 1996. 5, 40

[2] W. Ashmawi, R. Guerin, S. Wolf, and M. Pinson. On the impact of policing and rate guarantees in diff-serv networks: A video streaming application perspective. *Proceedings of ACM SIGCOMM'2001*, pages 83–95, 2001. 34

[3] H. Balakrishnan, H. Rahul, and S. Seshan. An Integrated Congestion Management Architecture for Internet Hosts. In *Proceedings of ACM SIGCOMM*, Cambridge, MA, USA, Sept. 1999. 2, 20, 21, 170

[4] J. Beerends and J. Stemerdink. A perceptual speech quality measure based on a psychoacoustic sound representation. *Journal of Audio Eng. Soc.*, 42:115–123, Dec. 1994. 9

[5] U. Benzler. Spatial Scalable Video Coding Using a Combined Subband-DCP Approach. *IEEE Transaction on Circuits and System for Video Technology*, 10:1080–1087, Oct. 2000. 43

[6] P. Bocheck, A. Campbell, S.-F. Chang, and R. Lio. Utility-based Network Adaptation for MPEG-4 Systems. In *Proceedings of Workshop on Network*

and Operating System Support for Digital Audio and Video (NOSSDAV), Basking Ridge, NJ, USA, June 1999. 3

[7] J.-C. Bolot, S. Fosse-Parisis, and D. Towsley. Adaptive FEC-Based Error Control for Internet Telephony. In *Proceedings of IEEE INFOCOM*, New York, NY, Mar. 1999. 4, 5, 17

[8] J.-C. Bolot and T. Turletti. Adaptive Error Control for Packet Video in the Internet. In *Proceedings of IEEE International Conference on Image Processing (ICIP)*, Lausanne, Switzerland, Sept. 1996. 40

[9] C. Bourassa, M. Figura, and M. Scaviola. Adaptive Forward Error Correction for MPEG Streaming Video. *Major Qualifying Project MQP-MLC-AF03*, Worcester Polytechnic Institute, May 2004. Advisors Mark Claypool and Robert Kinicki. 160, 161

[10] J. Boyce and R. Gaglianello. Packet Loss Effects on MPEG Video sent over the Public Internet. In *Proceedings of ACM Multimedia*, pages 181–190, Bristol, U.K., Sept. 1998. 4

[11] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, K. R. L. Peterson, S. Shenker, J. Wroclawski, and L. Zhang. Recommendations on Queue Management and Congestion Avoidance in the Internet. *IETF Request for Comments (RFC) 2309*, Apr. 1998. 21

[12] G. Carle and E. W. Biersack. Survey of Error Recovery Technologies for IP-based Audio-Visual Multicast Applications. *IEEE Network*, 1997. 40

[13] M. Carson and D. Santay. NIST Net - A Linux-based Network Emulation Tool. URL: http://snad.ncsl.nist.gov/itg/nistnet. 163

[14] S. Cho. Adaptive Error Control for Hybrid (Satellite-Terrestrial) Networks. In *Proceedings of IEEE Wireless Communications and Networking Conference*, pages 1013–1017, New Orleans, LA, Sept. 1999. 39

[15] J. Chung, M. Claypool, and Y. Zhu. Measurement of the Congestion Responsiveness of RealPlayer Streaming Video Over UDP. In *Proceedings of Packet Video (PV) Workshop*, Nantes, France, Apr. 2003. 4, 68, 79, 97, 111, 150, 152, 153, 166

[16] G. J. Conklin and S. S. Hemami. A Comparison of Temporal Scalability Techniques. *IEEE Transaction on Circuits and System for Video Technology*, 9:909–919, Sept. 1999. 43

[17] B. Dempsey, T. Strayer, and A. Weaver. Adaptive Error Control for Multimedia Data Transfers. In *Proceedings of International Workshop on Advanced Teleservices and High-Speed Communication Architectures (IWACA)*, pages 279–289, Munich, Germany, Mar. 1992. 37

[18] T. DeSantis and D. Loose. TCP Traffic Analysis. *Major Qualifying Project MQP-MLC-MT03*, Worcester Polytechnic Institute, Sept. 2003. Advisors Mark Claypool and Robert Kinicki. 3

[19] M. Domanski, A. Luczak, and S. Mackowiak. Spatial-temporal Scalability for MPEG Video Coding. *IEEE Transaction on Circuits and System for Video Technology*, 10:1088–1093, Oct. 2000. 43

[20] A. Dumitras and B. G. Haskell. I/P/B Frame Type Decision by Collinearity of Displacements. In *Proceedings of IEEE International Conference on Image Processing (ICIP)*, Singapore, Oct. 2004. 62, 67

[21] N. Feamster and H. Balakrishnan. Packet Loss Recovery for Streaming Video. In *Proceedings of Packet Video (PV) Workshop*, Pittsburgh, PA, USA, Apr. 2002. 37, 78, 97

[22] N. Feamster, D. Bansal, and H. Balakrishnan. The Interactions between Layered Quality Adaptation and Congestion Control for Streaming Video. In *Proceedings of Packet Video (PV) Workshop*, Kyongju, Korea, Apr. 2001. 43

[23] W.-C. Feng, J. Choi, W.-C. Feng, and J. Walpole. Under the Plastic: A Quantitative Look at DVD Video Encoding and Its Impact on Video Modeling. In *Proceedings of Packet Video (PV) Workshop*, Nantes, France, Apr. 2003. 62, 66

[24] S. Floyd. TCP and Explicit Congestion Notification. *ACM Computer Communication Review*, 24(5):10 – 23, Oct. 1994. 3

[25] S. Floyd and K. Fall. Promoting the Use of End-to-End Congestion Control in the Internet. *IEEE/ACM Transactions on Networking*, Feb. 1999. 21, 104

[26] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-Based Congestion Control for Unicast Applications. In *Proceedings of ACM SIGCOMM*, pages 43 – 56, Stockholm, Sweden, Aug. 2000. 2, 20, 21, 150, 170

[27] P. Frossard and O. Verscheure. Joint Source/FEC Rate Selection for Quality-Optimal MPEG-2 Video Delivery. *IEEE Transactions on Image Processing*, 10(12):1815–1825, Dec. 2001. 92, 104

[28] M. Fumagalli, R. Lancini, and S. Tubaro. A Novel Error-Concealment Algorithm for an Unbalanced Multiple Description Coding Architecture.

In *Proceedings of Packet Video (PV) Workshop*, Irvine, California, USA, Dec. 2004. 41

[29] I.-T. R. G.107. The e-model, a computational model for use in transmission planning. 9

[30] D. L. Gall. MPEG: A Video Compression Standard for Multimedia Applications. *Communications of the ACM*, 34(4):46 – 58, 1991. viii, 28, 65

[31] M. W. Garret and W. Willinger. Analysis, Modeling and Generation of Self-Similar VBR Video Traffic. In *Proceedings of ACM SIGCOMM*, Aug. 1994. 159

[32] M. Ghanbari. *Standard Codecs: Image Compression to Advanced Video Coding*. Institution Electrical Engineers, 2003. ISBN 0852967101. 16

[33] B. Girod, N. Farber, and E. Steinbach. Error-Resilient Coding for H.263. In *Insights into Mobile Multimedia Communication*, New York, NY, 1999. 41

[34] V. Hardman, M. A. Sasse, M. Handley, and A. Watson. Reliable Audio for Use over the Internet. In *Proceedings of Internet Society's International Networking Conference (INET)*, Ohahu, Hawaii, USA, June 1995. 5, 78

[35] F. Hartanto and H. R. Sirisena. Hybrid Error Control Mechanism for Video Transmission in the Wireless IP Networks. In *Proceedings of the Tenth IEEE Workshop on Local and Metropolitan Area Networks (LANMAN)*, Sydney, Australia, Nov. 1999. 78, 97

[36] International Telecommunications Union. One-Way Transmission Time. Technical Report G.114, ITU-T Recommendation, 1996. 37

[37] International Telecommunications Union. Methodology for the Subjective Assessment of the quality of Television Pictures. Technical Report ITU-R BT.500-11, ITU Recommendation, 2002. 33

[38] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley. Inferring TCP Connection Characteristics Through Passive Measurements. In *Proceedings of IEEE INFOCOM*, Hong Kong, China, Mar. 2004. 68, 79, 97, 111, 166

[39] S.-R. Kang and D. Loguinov. Impact of FEC Overhead on Scalable Video Streaming. In *Proceedings of Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, Stevenson, Washington, USA, June 2005. 39

[40] H. Katata, N. Ito, and H. Kusao. Temporal-scalable Coding Based on Image Content. *IEEE Transaction on Circuits and System for Video Technology*, 7:52–59, Feb. 1997. 43

[41] C. Krasic, J. Walpole, and W.-C. Feng. Quality-adaptive Media Streaming by Priority Drop. In *Proceedings of Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, pages 112–121, Monterey, CA, USA, 2003. 30, 45, 104, 105

[42] M. Krunz, R. Sass, and H. Hughes. Statistical Characteristics and Multiplexing of MPEG Streams. In *Proceedings of IEEE INFOCOM*, pages 455–462, Boston, MA, Apr. 1995. 79

[43] J. Y. Lee and H. Radha. Interleaved Source Coding (ISC) for Predictive Video over ERASURE-Channels. In *Proceedings of Packet Video (PV) Workshop*, Irvine, California, USA, Dec. 2004. 38

[44] W. Li. Overview of Fine Granularity Scalability in MPEG-4 Video Standard. *IEEE Transaction on Circuits and System for Video Technology*, 2001. 29

[45] R. R.-F. Liao and A. T. Campbell. A Utility-Based Approach for Quantitative Adaptation in Wireless Packet Networks. *Wireless Networks*, 7(5):541–557, 2001. 34

[46] Y. Liu. Video Redundancy – A Best-Effort Solution to Network Data Loss. Master's thesis, Worcester Polytechnic Institute, May 1999. Advisor: Mark Claypool. 40

[47] Y. Liu and M. Claypool. Using Redundancy to Repair Video Damaged by Network Data Loss. In *Proceedings of IS&T/SPIE/ACM Multimedia Computing and Networking (MMCN)*, Jan. 2000. 4, 78, 97, 104

[48] D. Loguinov and H. Radha. Measurement Study of Low-bitrate Internet Video Streaming. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop (IMW)*, San Francisco, California, USA, Nov. 2001. 150, 151

[49] R. Mahajan, S. Floyd, and D. Wetherall. Controlling High-Bandwidth Flows at the Congested Router. In *Proceedings of the 9th International Conference on Network Protocols (ICNP)*, Riverside, CA, USA, Nov. 2001. 2

[50] K. Mayer-Patel, L. Le, and G. Carle. An MPEG Performance Model and Its Application To Adaptive Forward Error Correction. In *Proceedings of ACM Multimedia*, Juan Les Pins, France, December 2002. 5, 39, 45, 62, 66

[51] J. D. McCarthy, M. Sasse, and D. Miras. Sharp or Smooth? Comparing the Effects of Quantization vs. Frame Rate for Streamed Video. In *Proceedings of ACM CHI 2004 Conference on Human Factors in Computing Systems*, Vienna, Austria, Apr. 2004. 43

[52] A. Mena and J. Heidemann. An Empirical Study of Real Audio Traffic. In *Proceedings of IEEE INFOCOM*, pages 101 – 110, Tel-Aviv, Israel, Mar. 2000. 2

[53] Microsoft Corporation. Media Player 10, Copyright 2004. URL:http://www.microsoft.com/windows/windowsmedia/mp10. 44

[54] J. Mitchell and W. Pennebaker. *MPEG Video: Compression Standard*. Chapman and Hall, 1996. ISBN 0412087715. viii, 4, 14, 28

[55] M. Miyabayashi, N. Wakamiya, M. Murata, and H. Miyahara. MPEG-TFRCP: Video Transfer with TCP-Friendly Rate Control Protocol. In *Proceedings of IEEE International Conference on Communications (ICC)*, pages 137–141, St. Petersburg, Russia, 2001. 43

[56] T. Naveen and J. W. Woods. Motion Compensated Multiresolution Transmission of High Definition Video. *IEEE Transaction on Circuits and System for Video Technology*, 4:29–41, Feb. 1994. 43

[57] T. Nguyen and A. Zakhor. Distributed Video Streaming with Forward Error Correction. In *Proceedings of Packet Video (PV) Workshop*, Apr. 2002. 4

[58] J. Nichols. Measurement of Windows Streaming Media. Master's thesis, Worcester Polytechnic Institute, Feb. 2004. Advisor: Mark Claypool and Robert Kinicki. 2, 21

[59] U. of California Berkeley. Berkeley MPEG-2 Encoder and Player. Internent site http://bmrc.berkeley.edu/frame/research/mpeg/. 98

[60] C. Padhye, K. Christensen, and W. Moreno. A New Adaptive FEC Loss Control Algorithm for Voice Over IP Applications. In *Proceedings of IEEE International Performance, Computing and Communication Conference*, Phoenix, AZ, Feb. 2000. 4, 5

[61] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and Its Empirical Validation. In *Proceedings of ACM SIGCOMM*, Vancouver, Brisish Columbia, Candada, 1998. 20, 50, 97, 125

[62] K. Park and W. Wang. QoS-Sensitive Transport of Real-Time MPEG Video Using Adaptive Forward Error Correction. In *Proceedings of IEEE Multimedia Systems*, pages 426 – 432, Florence, Italy, June 1999. 4, 5

[63] V. Paxson. End-to-End Internet Packet Dynamics. *IEEE/ACM Transactions on Networking*, 7(3):277–292, June 1999. 79, 150

[64] C. Perkins, O. Hodson, and V. Hardman. A Survey of Packet-Loss Recovery Techniques for Streaming Audio. *IEEE Network Magazine*, Sep/Oct 1998. viii, 17, 36, 37

[65] M. Pinson and S. Wolf. A New Standardized Method for Objectively Measuring Video Quality. *IEEE Transactions on Broadcasting*, 50(3):312–322, Sept. 2004. 32

[66] M. Pinson and S. Wolf. A New Standardized Method for Objectively Measuring Video Quality. *IEEE Transactions on Broadcasting*, 50(3):312–322, Sept. 2004. 91

[67] C. Podilchunk, N. Jayant, and N. Farvardin. Three dimensional subband coding of video. *IEEE Transactions on Image Processing*, 4:125–139, Feb. 1995. 43

[68] J. Qiu and J. Mark. Error Control for Integrated Wireless and Wireline Networks. In *Proceedings of Conference on Information Sciences and Systems (CISS)*, Princeton, NJ, Mar. 1998. 41

[69] Real Networks Incorporated. RealPlayer 8 User Manual, copyright 2000. URL: http://service.real.com/help/player/plus_manual.8-/rppmanual.htm. 44

[70] Real Networks Incorporated. RealProducer User's Guide, copyright 2000. URL: http://www.service.real.com/help/library/guides/producerplus85-/producer.htm. 80

[71] Real Networks Incorporated. RealVideo 10 Technical Overview, Copyright 2004. URL: http://docs.real.com/docs/rn/rv10-/RV10_Tech_Overview.pdf. 44

[72] I. S. Reed and G. Solomon. Polynomial Codes Over Certain Finite Fields. *Journal of the Society of Industrial and Applied Mathematics (SIAM)*, 8(2):300–304, June 1960. 17

[73] R. Rejaie, M. Handley, and D. Estrin. Quality Adaptation for Congestion Controlled Video Playback over the Internet. In *Proceedings of SIGCOMM*, pages 189–200, Cambridge, MA, Aug. 1999. 43

[74] R. Rejaie, M. Handley, and D. Estrin. RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet. In *Proceedings of IEEE INFOCOM*, pages 1337 – 1345, New York, NY, Mar. 1999. 2, 20, 21, 170

[75] I. Rhee. Error Control Techniques for Interactive Low-Bit Rate Video Transmission over the Internet. In *Proceedings of ACM SIGCOMM*, Vancouver, British Columbia, Canada, Sept. 1998. 41, 78, 97

[76] A. Rix. Advances in objective quality assessment of speech over analogue and packet-based networks. In *IEEE Data Compression Colloquium*, London, UK, Nov. 1999. 9

[77] L. Rizzo. Effective Erasure Codes for Reliable Computer Communication Protocols. *Computer Communication Review*, Apr. 1997. 163

[78] O. Rose. Statistical Properties of MPEG Video Traffic and their Impact on Traffic Modeling in ATM Systems. In *Proceedings of the Conference of Local Computer Networks*, pages 397–406, Minneapolis, MN, USA, Oct 1995. 154

[79] S. Sakazawa, Y. Takishima, M. Wada, and Y. Hatori. Coding Control Scheme for a Multi-Encoder System. In *Proceedings of Packet Video (PV) Workshop*, Brisbane, Australia, Mar. 1996. 92

[80] W.-T. Tan and A. Zakhor. Real-Time Internet Video Using Error Resilient Scalable Compression and TCP-Friendly Transport Protocol. *IEEE Transactions on Multimedia*, 1(2):172–186, 1999. 45

[81] D. Taubman and A. Zakhor. Multirate 3-D Subband Coding of Video. *IEEE Transactions on Image Processing*, 3:572–588, Sept. 1994. 43

[82] A. Tripathi and M. Claypool. Improving Multimedia Streaming with Content-Aware Video Scaling. In *Workshop on Intelligent Multimedia Computing and Networking (IMMCN)*, Durham, North Carolina, USA, Mar. 2002. 3

[83] Y. Wang, M. Claypool, and Z. Zuo. An Empirical Study of RealVideo Performance Across the Internet. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop (IMW)*, pages 295 – 309, San Francisco, California, USA, Nov. 2001. 2

[84] S. Winkler and C. Faller. Audiovisual quality evaluation of low-bitrate video. In *Proceedings of SPIE Human Vision and Electronic Imaging*, San Jose, CA, Jan. 2005. 34

[85] H. Wu, M. Claypool, and R. Kinicki. A Model for MPEG with Forward Error Correction and TCP-Friendly Bandwidth. In *Proceedings of Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, Monterey, CA, USA, June 2003. 5

[86] H. Wu, M. Claypool, and R. Kinicki. Adjusting Forward Error Correction with Temporal Scaling for TCP-Friendly Streaming MPEG. Technical Report WPI-CS-TR-03-10, CS Department, Worcester Polytechnic Institute, Apr. 2003. 79, 84

[87] H. Wu, M. Claypool, and R. Kinicki. Adjusting Forward Error Correction with Quality Scaling for Streaming MPEG. In *Proceedings of Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, Stevenson, Washington, USA, June 2005. 104

[88] H. Wu, M. Claypool, and R. Kinicki. Adjusting Forward Error Correction with Temporal Scaling for TCP-Friendly Streaming MPEG. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 1(4), Nov. 2005. 104

[89] H. Wu, M. Claypool, and R. Kinicki. Guidelines for Selecting Practical MPEG Group of Pictures. Technical Report WPI-CS-TR-05-18, CS Department, Worcester Polytechnic Institute, Aug. 2005. 57, 63

[90] H. Wu, M. Claypool, and R. Kinicki. A Study of Video Motion and Scene Complexity Characteristics. Technical Report WPI-CS-TR-06-04, CS Department, Worcester Polytechnic Institute, May 2006. 65

[91] Y. Yokoyama. Adaptive GOP Structure Selection for Real-time MPEG-2 Video Encoding. In *Proceedings of IEEE International Conference on Image Processing (ICIP)*, Vancouver, Canada, Sept. 2000. 62, 67

[92] Y. Zhu. Using Interleaving to Ameliorate the Effects of Packet Loss in a Video Stream. Master's thesis, Worcester Polytechnic Institute, June 2001. Advisor: Mark Claypool. 38