Simulation Tool and Metric for Evaluating Wireless Digital Video Systems

Pablo Maximiliano Robert

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical and Computer Engineering

Jeffrey H. Reed, Chair
Brian D. Woerner
Willard Farley, Jr.

September 23, 1998
Blacksburg, Virginia

Keywords: Video, LMDS, RSVP, Wireless Video, Simulation

Simulation Tool and Metric for Evaluating Wireless Digital Video Systems

Pablo Maximiliano Robert

(ABSTRACT)

This thesis focuses on the interaction between digital video quality and channel coding in a wireless communication system. Digital video is a high-bandwidth, computationally intensive application. The recent allocation of large tracks of spectrum by the FCC has made possible the design and implementation of personal wireless digital video devices for several applications, from personal communications to surveillance. Wireless video research has centered on contextual coding mechanisms; different sections of a video data stream are coded differently based on the perceived importance of the coded bits. Besides the complexity introduced into a system by this type of coding, no metric based solely on physical layer parameters can be used to predict video quality across different system architectures. A tool was built to explore the video/channel coding relationship. This tool simulates a packet-based digital wireless transmission in various noise and interference environments. The basic communications system models the DAVIC (Digital Audio-Visual Council) layout for the LMDS (Local Multipoint Distribution Service) system and includes several error control algorithms and a packetizing algorithm that is MPEG-compliant. This work introduces a statistical approach to monitoring the video quality. The error-event-mean-arrival-rate, $\lambda_{distance}$, is a metric derived from the physical layer that can be used to predict the expected video quality across systems with different channel coding algorithms. This metric proved to be uniformly consistent in predicting video quality for sequences corrupted by Gaussian and non-Gaussian noise and protected by a variety of error correction techniques.

ACKNOWLEDGEMENTS

I would like to express my most sincere appreciation and deep gratitude to my advisor, Dr. Jeffrey H. Reed. Not only has his knowledge of wireless communications as a whole been of invaluable help, but also his support has made my stay at Virginia Tech possible. I would also like to thank Dr. Ahmed Darwish, whose guidance, especially in the difficult early stages of the project, has made this work possible. I would also like to thank Dr. Brian D. Woerner and Willard Farley, Jr. in their support as committee members.

I also owe a great deal of thanks to my parents, Edgardo and Elba Robert. I have been blessed with their continuous support through my undergraduate and graduate programs. It is in large part through their support that I strive to reach my potential. I would also like to thank both my sisters, Natalia and Cecilia, whose thoughtfulness and kindness has brought a smile to my face on many occasions over the years.

I would also like to thank Lori Hughes for applying her technical editing skills to my work.

Finally, I would like to thank my colleagues and friends, both on- and off-campus, who have made my stay at Virginia Tech an enjoyable and rewarding personal experience.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Justification

In recent years, interest in digital video transmissions has increased dramatically. The basis for this interest rests on two events: the creation of small fast computers and the allocation of vast stretches of radio spectrum. The advent of small, inexpensive, and fast computers allows the average user to decode digital video real-time. Until recently, this ability rested only with users capable of spending large amounts of money on dedicated hardware. This availability of inexpensive video-decoding hardware is evident in the mushrooming direct satellite broadcast television services. In addition, large tracks of spectrum, such as the Ku band for satellite transmissions, the Local Multipoint Distribution Service (LMDS) band at 28 GHz, and the 38 GHz band have opened, allowing the transmission of vast amounts of information. These spectrum allocations allow space- and terrestrial-based systems to be deployed that carry large amounts of information, such as digital video.

1.2 Problem Definition

When designing a geo-synchronous satellite broadcast system, factors such as frequency reuse are minor problems. The system layout is generally based on a small number of servers (satellites) that

service a very large area, such as the continental United States. On terrestrial systems, on the other hand, such problems as Line-of-Sight (LOS) blockage and capacity limitations in interactive systems limit the server size to a small area in microwave-based systems. In these systems, interference from co-channel sources becomes a problem that needs to be analyzed.

Another problem in terrestrial radio systems is the time-variant radio channel, which can lower the received signal strength below acceptable levels. This time-variant channel may be due to a moving transmitter or receiver, or it may also be due to the multipath characteristics of a channel changing due to a dynamic environment, such as next to a road with traffic. The effect of low SNR on a video transmission is a topic that has been studied in the past with emphasis on the video encoding/decoding process. Researchers have placed emphasis on improving system performance by changing levels of error correction power as a function of the contextual value of the bits being transmitted; these dynamic solutions respond to real-time video needs and require a real-time evaluation of the video being transmitted at the channel level. This approach is cumbersome and very complicated since it implies that the physical layer of the communications system needs to understand the data being transmitted and it needs to be quickly reconfigured. Given the complexity of this approach, it has led some researchers to claim that "[This] problem is very difficult, if not impossible, to solve due to the many involved variables and the fact that it is often difficult to model or describe these variables."[2] This paper will investigate this issue from the communications system side of the problem. In particular, a simplified statistical method for evaluating expected system performance is proposed and analyzed. This method is validated by simulating several systems with different channel coding algorithms.

One of the key challenges of a real-time configurable system is to correctly assess the current environmental conditions and how they pertain to different system architectures. It is important to correctly assess the environment in which the system is operating to offer the end-user the best possible service with minimal spectral resources. Determining the quality of video from the channel conditions has been an elusive goal. The availability of a simple statistical approach to model the channel conditions as they pertain to system performance opens the door to simple adaptive video algorithms that were not possible before. The key is simplicity; digital video is a computationally-intensive application, and reducing the overhead inherent to a digital video system can simplify the

hardware layout needed to implement a personal wireless digital video system so that it is realizable given the current level of technology.

1.3 Summary of Approach

The core of this project is a simulation tool designed to investigate video transmissions in an LMDS system, the Reconfigurable Simulation of Video Performance (RSVP). RSVP is designed to simulate a system built around the guidelines presented by DAVIC (Digital Audio-Visual Council) that illustrate a basic system layout for several technologies. DAVIC presents today's dominant system architecture for LMDS systems. Different configurations of the DAVIC-based system will be simulated to investigate the effect of noise and co-channel interference on video quality. Only a small fraction of the capabilities of this simulation were used in this thesis. The simulation tool is flexible enough to allow the study of a wide range of environments and conditions, and it can serve as a key tool in future efforts in this field.

A statistical approach to predicting video quality based on physical layer parameters is presented. The error-event-mean-arrival-rate, $\lambda_{distance}$, is a metric derived from the physical layer that can be used to predict the expected video quality across systems with different channel coding algorithms. This metric proved to be uniformly consistent in predicting video quality for sequences corrupted by Gaussian and non-Gaussian noise and protected by a variety of error correction techniques.

1.4 Thesis Overview

Chapter 2 of this thesis will first review the necessary background for the project; the DAVIC system layout and the basics of the video coding part of the MPEG-2 standard. Chapter 3 reviews the capabilities of the simulation software developed and the validation of individual modules. Chapter 4 presents test scenarios and results. Chapter 5 analyzes the data presented in Chapter 4 and introduces the simplified physical layer-based video metric, including possible applications for the metric. Chapter 6 suggests further avenues of research. Several appendices cover points concerning this thesis that were not directly relevant to the presented research. Appendix A discusses the

packaging of MPEG video in an MPEG-2/Transport-Stream packet format. Appendix B covers the simulation architecture and user interface. Appendix C reviews the random number generation algorithms used in the simulation.

Chapter 2

Background

Several aspects of video and telecommunications, physical layer system design, video coding, data packet construction, and channel modeling need to be addressed to provide the proper background for later chapters. Once the appropriate groundwork is laid, the system simulation can be explained and the results can be interpreted.

2.1 Physical Layer Layout

LMDS is a new service that has yet to be offered to the general public. For this reason, there are currently no set industry standards. However, there exists a set of guidelines presented by the Digital Audio-Visual Council (DAVIC) that is widespread across industry. These guidelines are intended to allow inter-operability between the equipment of different vendors. The guidelines are limited to the physical layer of the system, and the network layout is left up to the designer's discretion. To make the simulation relevant to current systems, the basic layout for the LMDS system that was simulated follows the DAVIC guidelines. DAVIC version 1.3, part 8, section 7.11 describes the physical layer of the LMDS system simulated. For the purposes of this thesis, the DAVIC guidelines for the downlink and uplink can be summarized as follows [3].

2.1.1 Downlink

Figure 2.1 shows the layout for the downlink transmitter and receiver sets. Each module that must be implemented in the DAVIC-compliant system is discussed in some detail.

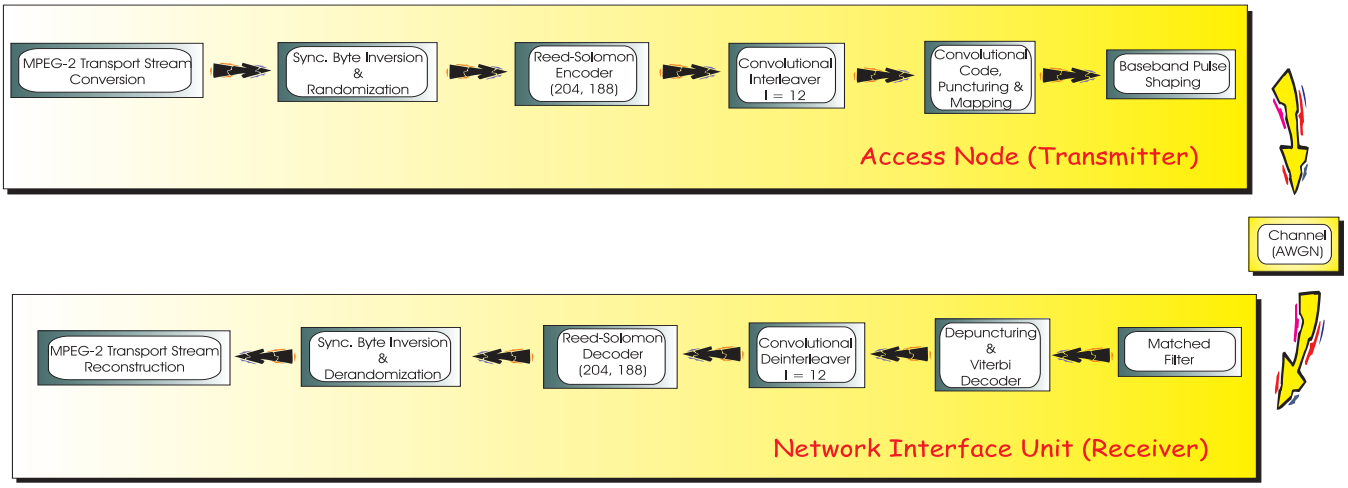

Figure 2.1: DAVIC-Compliant LMDS Downlink

Data Layout

The system is a packet-driven system. All data must be packed in 188-byte-long packets. The format for the data within the packet depends on the type of data. This data may take two formats, video or ATM data. The format for the data packing is discussed in detail in the Packet Data section of the paper.

Randomizer

A randomizer, based on a PN sequence generator, spreads the signal. Figure 2.2 shows the layout of the randomizer. The shift register has a generator function of $1 + x^{14} + x^{15}$, with an initialization of 0x4A80. The randomizer is initialized every time it encounters the inverted packet header (0xB8). The packet header is not randomized. The reason why a randomizer is used is to increase the number of transitions in the data stream to help the receiver synchronize to and acquire the received symbols.

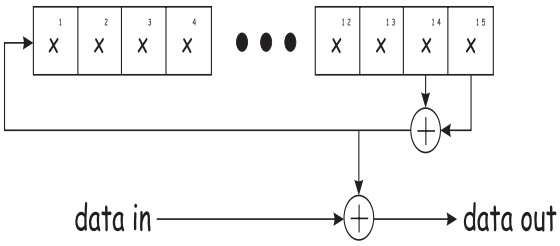

Figure 2.2: Randomizer/Derandomizer Schematic

Packet Initialization

Initialization is performed on every packet (0x47). Every eighth packet is inverted (0xB8) to facilitate synchronization. This header is the first byte of each 188-byte packet. Because the header must be present in every packet transmitted, the payload of each packet is 187 bytes.

Error Correction

A concatenated code is used for error correction. This code is composed of a Reed-Solomon (RS) code back-to-back with a convolutional code. The convolutional code has a constraint length of seven with a base rate of one-half. The generator polynomials for each of the two paths is 171_{oct} and 133_{oct}. Figure 2.3 illustrates a schematic of the convolutional encoder used. This convolutional encoder is a variable rate encoder. A puncture map (Table 2.1) is provided that shows the punctures used to achieve different encoding rates.

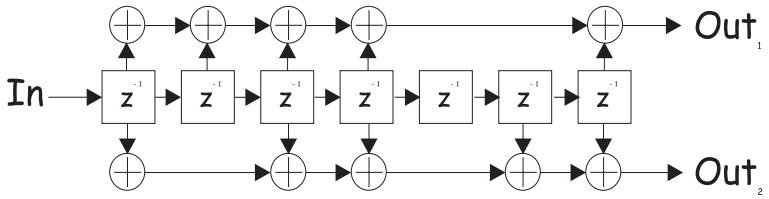

Figure 2.3: $R = \frac{1}{2}$ Convolutional Encoder

Figure 2.4 presents a diagram of the circuit used for the puncturing process, where a certain number of symbols are eliminated to increase the effective rate of the code. X-control and Y-control are registers listed as X and Y in Table 2.1; a zero means the switch is open, and a one means the switch

Table 2.1: Variable Rate Convolutional Encoder Puncture Map

Code Rates	X	Y	I	Q
$\frac{1}{2}$	1	1	X_1	Y_1
$\frac{2}{3}$	10	11	$X_1Y_2Y_3$	$Y_1X_3Y_4$
$\frac{3}{4}$	101	110	X_1Y_2	Y_1X_3
$\frac{5}{6}$	10101	11010	$X_1Y_2Y_4$	$Y_1X_3X_5$
$\frac{7}{8}$	1000101	1111010	$X_1Y_2Y_4Y_6$	$Y_1Y_3X_5X_7$

is closed. Once the incoming data has been punctured, a module eliminates the spots with no data (also controlled by X and Y). Once the punctured data has been eliminated from the stream, the resulting stream is modulated, resulting in the baseband I and Q signals.

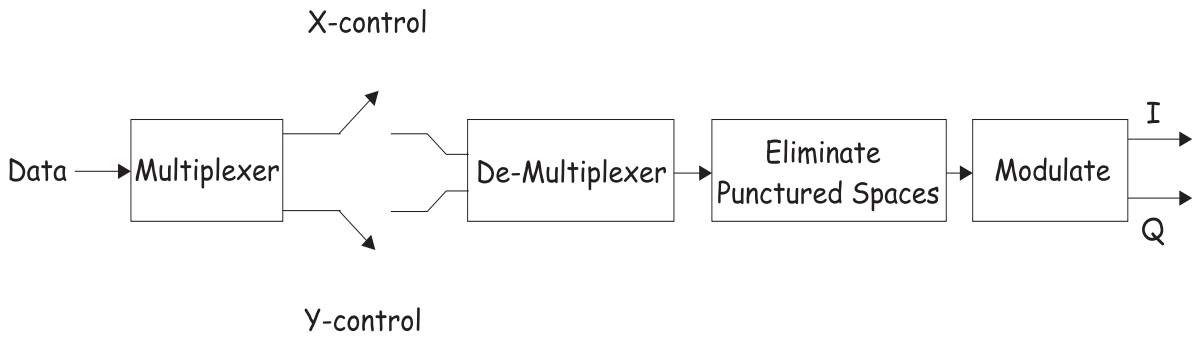

Figure 2.4: Puncturing Circuit

Decoding of the convolutional code is done with a Viterbi algorithm. The DAVIC standard does not specify whether soft or hard decision should be used to decode the message.

The RS code has a correction power of eight symbols and a (204,188) code characteristic. The Galois field generator polynomial is $x^8 + x^4 + x^3 + x^2 + 1$.

The gain attained from the combination of these two codes is far greater than the gain from each

code individually added together. The extra gain occurs because the correction characteristics of one coding method complements the correction characteristics of the other method. The main source of errors in a concatenated code can be expressed by the first-error-event probability of the code. The first-error-event probability of the code is the probability that an error will occur such that the subsequent correct symbols will yield a lower distance than the correct sequence. The error path will eventually converge with the original path. This divergence and convergence is the source of errors in the decoder. These errors, although sometimes long, are fairly evenly distributed along the sequence. The end result is a series of bursts of noise that are evenly spread. A RS code, on the other hand, corrects several errors that are at least one symbol (in this case, eight bits) long. This means that short bursts of errors committed by the convolutional code are treated as a single (or sometimes longer) byte error by the RS code. Clumping of several errors in a convolutional code is seen by the RS code as a single error committed by a single symbol. Whereas the convolutional code would have had several bit errors within the space of a few symbols, these errors are corrected by the RS code. There is a limit to this relationship. If there are more symbol errors in the sequence than the RS code can correct (eight in this case), then the combination will fail and an error is committed by the decoder.

Interleaving

An interleaver is added to the system between the convolutional and Reed-Solomon codes. The interleaver has a depth of twelve. That means that bytes that are twelve symbols apart are interleaved. Figure 2.5 shows a diagram of the interleaver used. The initialization byte is not interleaved.

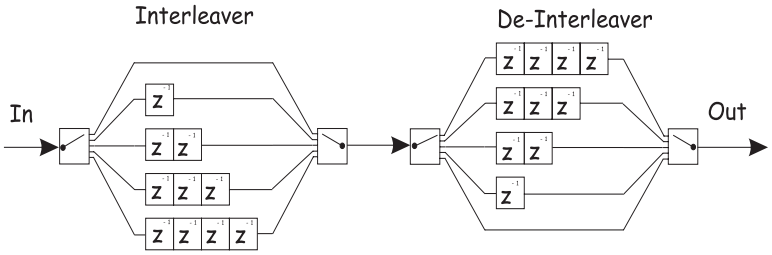

Figure 2.5: $I = 12$ Convolutional Interleaver

Figure 2.5 is a bit misleading: the length of the sequence does not change from interleaving. There

is no zero-padding data that has not made it through the M-stage FIFO. If the data is unavailable, the marker resets and the next set of bytes is analyzed. If a sequence would read X_1, X_2, X_3, X_4, ... , X_{n-2}, X_{n-1}, X_n, then the output would read X_1, X_{13}, X_2, X_{25}, X_{14}, X_3, ... , X_{n-2}, X_{n-13}, X_{n-24}, X_{n-1}, X_{n-12}, X_n.

Modulation

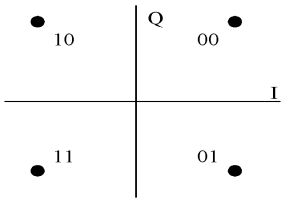

Figure 2.6: Downlink QPSK Constellation

The signal is modulated using QPSK (Figure 2.6) or 16-QAM (Figure 2.7). The convolutional encoder is not used if 16-QAM is used.

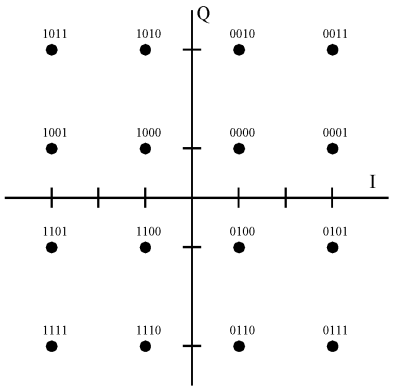

Figure 2.7: Downlink 16-QAM Constellation

Pulse Shaping and Spectral Mask

The pulses are then pulse-shaped using a root-raised cosine pulse-shaping filter. The Nyquist filters can be scaled to a roll-off of 0.2 or 0.35, depending on channel requirements. The receiver uses a root-raised cosine matched filter, yielding a Nyquist filter as the result. Equation 2.1 describes the

root-raised cosine filter's frequency response.

$$H(f) = \begin{cases} 1 & \text{for } |f| < f_N(1-\alpha) \\ \left\{\frac{1}{2} + \frac{1}{2}\sin\left(\frac{\pi}{2f_N}\left[\frac{f_N - |f|}{\alpha}\right]\right)\right\}^{\frac{1}{2}} & \text{for } f_N(1-\alpha) \leq |f| \leq f_N(1+\alpha) \\ 0 & \text{for } |f| > f_N(1+\alpha) \end{cases} . \tag{2.1}$$

The spectral mask is set in Table 2.2. The values are sectioned according to the filter roll-off (α) and f_N, the Nyquist sampling rate ($\frac{f_s}{2}$).

Table 2.2: Downstream Power Spectral Mask

$\frac{f-f_c}{f_N}$	QPSK Response [dB]	16QAM Response [dB]	Tolerance [dB]
$\leq 1-\alpha$	0	0	± 0.25
at 1	-3	-3	± 0.5
at $1+\alpha$	< -22	< -24	-
at $2-\alpha$	< -30	< -36	-
≥ 2	< -32	< -40	-

Channel

The channel is either 20 or 40 MHz wide. The downlink channels are to be separated by at least 20 MHz. If changed, this distance is increased in 1 MHz increments.

Antenna

Since the network layout is not discussed in this section, both the directivity and gain of the antenna are left to the user's discretion.

2.1.2 Uplink

The uplink for the system is considerably simpler than the downlink. Figure 2.8 shows a schematic of the uplink transmitter/receiver set.

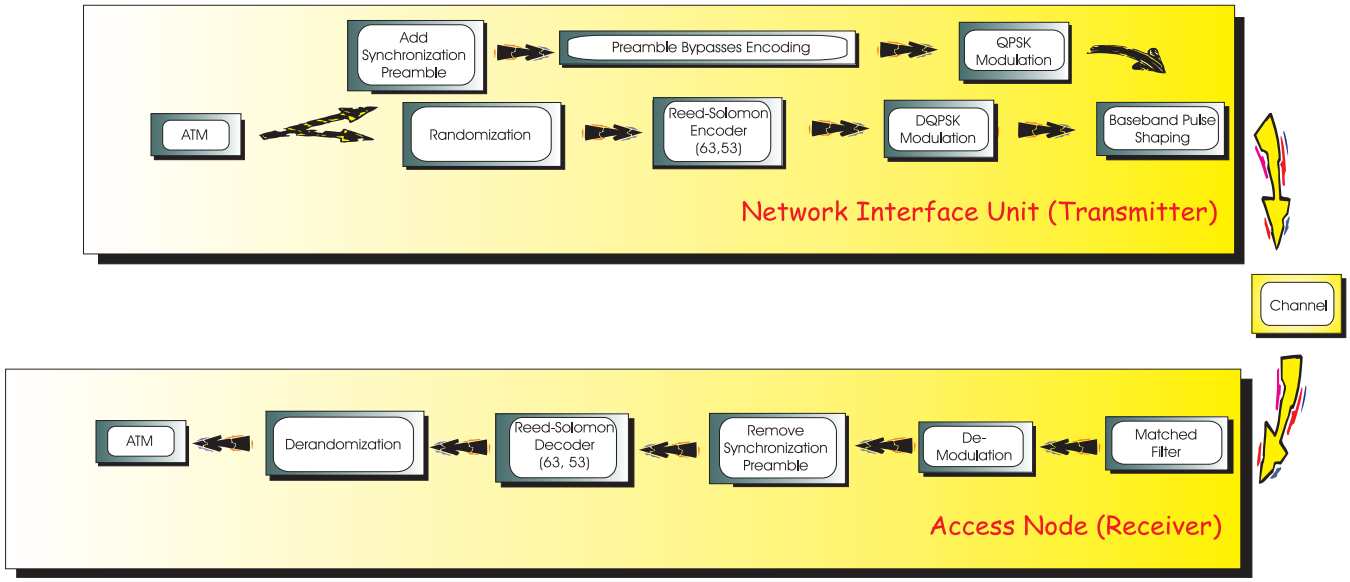

Figure 2.8: Uplink Transmitter/Receiver Schematic

The data end of the transmitter unit seen in Figure 2.8 is controlled by a decision switch. The only job of this switch is to detect the leading edge of a packet. Once a leading edge is detected, the top path in the transmit path is taken, and the preamble is modulated using QPSK. All other information is randomized, error-correction redundancy is added, and DQPSK is used for modulation.

Data Layout

The LMDS uplink is not expected to handle video transmissions. Although the system should be able to handle video conferencing (the data rate should be high enough), the bulk of information sent in the upstream is probably not video information. To that end, the uplink packet size is not MPEG-2 TS standard but ATM. Each packet that is handled by the upstream enters the system as a 53-byte ATM packet. ATM packing is discussed in detail in the Packet Data section of the

paper.

Preamble

Each packet is preceded by a four-byte preamble. The receiver synchronizes to the transmission by locking onto this preamble. The code for the preamble is 0x00FCFCF3.

Modulation

The four-byte preamble is modulated on a fixed QPSK constellation (see Figure 2.9). Each bit pair after the preamble is modulated using DQPSK. The DQPSK difference constellation is shown on Table 2.3.

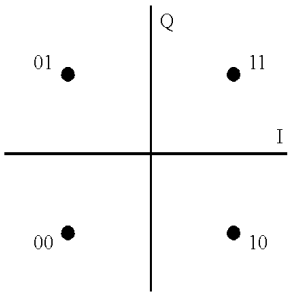

Figure 2.9: Uplink QPSK Constellation

Table 2.3: Uplink DQPSK Constellation

Bit 1	Bit 2	Phase Change
0	0	0^o
0	1	90^o
1	0	-90^o
1	1	180^o

Randomization

A randomizer with the same logical structure as the upstream (Figure 2.2) is used to randomize the signal. The generating polynomial used is $x^6 + x^5 + 1$. The six shift registers used in the randomizer are initialized to all ones.

Error Correction

A RS code is used as error protection. The RS code has a $x^8 + x^4 + x^3 + x^2 + 1$ Galois field generator polynomial. The RS code is (63,53); hence, it has a five-symbol correction capability.

Pulse Shaping

Like the downlink, the uplink uses a Nyquist filter for pulse shaping in the transmitter and for the matched filter in the receiver. Whereas the downlink uses a roll-off of either 0.20 or 0.35, the uplink uses a filter roll-off of 0.30.

Spectral Mask

The spectral mask follows the guidelines shown in Table 2.4. The values are sectioned according to the filter roll-off (α) and f_N, the Nyquist sampling rate ($\frac{f_s}{2}$).

Table 2.4: Upstream Power Spectral Mask

$\frac{f-f_c}{f_N}$	Response [dB]	Tolerance [dB]
$\leq 1-\alpha$	0	± 0.25
at 1	-3	± 0.5
at $1+\alpha$	< -22	-
at $2-\alpha$	< -30	-
≥ 2	< -32	-

The spectral mask for the uplink is identical to the spectral mask used in the downlink QPSK channel seen in Table 2.2.

Channel

The channel width depends on the grade selected for the communications link. Grade A uses channels 1-2.5 MHz wide, and Grade B uses channels 1-26 MHz wide. Channel spacing is at least 900 kHz with the separation increasing in 100 kHz increments.

Antenna

Since the network layout is not discussed in this section, both the directivity and gain of the antenna are left to the user's discretion.

2.2 MPEG

Currently, the most prominent digital video coding standard in the industry is the Motion Pictures Expert Group (MPEG) standard. MPEG is defined by the International Organization for Standardization (ISO) for the display of moving pictures and sound. MPEG is composed of several standards that are designed for particular applications, from low-end computer hardware to high-end studio equipment. It is important to understand the layout of the different standards and their goals. Since the goal of the project is to study the effect of a high-bit-error-rate environment on a video transmission, it is also important to have a basic understanding of the MPEG video coding algorithm and the format that the coded streams take when transmitted as a cohesive program.

2.2.1 Organization

MPEG was originally designed to view moving pictures through the use of a desktop computer. To that end, MPEG-1 (ISO-11172) was designed to offer video at relatively low speeds (under 1.5 Mbps). Computer storage was very expensive, and most computers had limited storage capacity.

In addition, the computers available to the general public had a low computational speed, so the low bit rate was a necessity.

Several events in the computer industry prompted additions to the MPEG standard. First, the average speed of microprocessors increased for a given purchase price; the average price of an instruction-per-second dropped considerably. In addition to the drop in price, the industry has kept up with Moore's prediction (doubling of computing power per unit area in a chip every eighteen months). Furthermore, the average price of memory and semi-permanent storage has also dropped significantly with storage capacity improving considerably for a constant price level.

These changes in the capabilities and price of computer hardware have allowed applications that were not possible before. Since performance increased so much, more complicated processing of larger amounts of data became possible. MPEG-2 (ISO-13818) is the resulting standard. MPEG-2 is backwards compatible with MPEG-1, meaning that an MPEG-1 sequence can be played by an MPEG-2 decoder. However, MPEG-2 was designed for an entirely different market than the target of MPEG-1: MPEG-2 was designed for the high-definition television market.

The two main differences between MPEG-1 and MPEG-2 are the ability to create a program in packet form and the ability of the decoder to use different parts of the incoming stream to fit computational limitations, available bandwidth, and user requirements. These differences are discussed in further detail in the video coding and program stream sections.

The structure of the text in the standard is the same for MPEG-1 and MPEG-2. Table 2.5 describes the structure of the document available from the ISO.

Since MPEG-2 is the standard used for high-definition television, it is the standard that will discussed in detail in the following sections. Parts one and two of the standard are discussed. Part one describes the format for the program information: how to pack information and send it over the packet network. Part two describes the video coding method used, which is helpful in understanding the effects of a communications channel on the video transmission.

Three more standards have surfaced since the conception of MPEG-2. MPEG-3 was conceived as a standard for very high data rate applications, such as high-definition television. During the development of MPEG-2, it was decided that MPEG-2 was scalable to meet the MPEG-3

Table 2.5: MPEG Standard Sections

Part	Description
1	Systems - Program/Transport Stream Layout
2	Video Sequence Coding
3	Audio Sequence Coding
4	Compliance Testing
6	Extensions for Digital Storage Media Command & Control
9	Extensions for Real Time Interface for System Decoders

requirements, so MPEG-3 was canceled. MPEG-4 was designed with very low bit rates in mind, far lower than MPEG-1; MPEG-4's design application is video conferencing applications. The latest addition to the MPEG standard is MPEG-7 (there is no MPEG-5 or MPEG-6), a searching standard. The idea is to create a standard that allows the user to search through a video library for particular material. This standard is based on the MPEG-2 and MPEG-4 program information structure. MPEG-7 is in the planning stages and will be phased in over the next few years.

2.2.2 Video Coding

To understand the effects of data errors on a video sequence, it is essential to understand the way that video is coded. A video sequence is defined as a single video clip. There is no audio attached to this elementary sequence. It is a single sequence not packed into any program structure. The sequence describes the encoding of a single video sequence. The section on program structure describes in detail the process of combining several video streams into one or more full programs.

Parts two of both ISO-11172 and ISO-13818 [4] describe the way MPEG codes video at relatively high data rates (higher than MPEG-4 rates). MPEG-2 is derived from MPEG-1 and is backward-compatible. For all practical purposes, MPEG-1 is a subset of MPEG-2. This discussion of video coding is based entirely on MPEG-2 although all the earlier coding information will also apply to MPEG-1.

Images are generated by creating a two-dimensional matrix that contains a set of points, also known as pixels. Each of these pixels is associated with a value that is interpreted as a particular color. Most images contain several same-colored pixels in several regions. This characteristic of images is called spatial redundancy. In other words, there are several objects that occupy large portions of the image but contain few or no features, such as an image of the sky. This redundancy can be greatly reduced through the use of a compression technique like the Huffman or Lempel-Ziv algorithms.

Moving pictures are composed of still images shown in rapid succession. The eye is a low-pass filter that will ignore the sharp transition between frames, and if the succession of images is fast enough, the illusion of motion is created. Because moving pictures consist of a series of still images, they have the redundancy available in still images. This redundancy is compounded by temporal redundancy, the high level of correlation between consecutive frames. In other words, a typical set of frames will present a series of images that are different in only a very few details. MPEG takes advantage of these two levels of redundancy to greatly reduce the amount of information that must be stored and transmitted.

Frame Structure

There are two types of compression, spatial and temporal. Spatial compression is based entirely on information available on a single frame. Temporal compression is based on comparisons between neighboring frames. Frames in MPEG are classified into three categories, each using different combinations of the two compression techniques.

Intra-coded (I) frames are reference frames that are composed entirely of spatial compression. Predictive-coded (P) frames are codes that perform a motion prediction algorithm based on the previous I or P frame. Bidirectionally predictive-coded (B) frames do motion prediction on the previous and next I or P frames.

Figure 2.10 illustrates the order in which a set of frames may be seen when decoded. Each I frame seen is a reference frame that has information that is encoded based entirely on information in that frame only and is used as a reference frame for subsequent B and P frames. The P frame is encoded

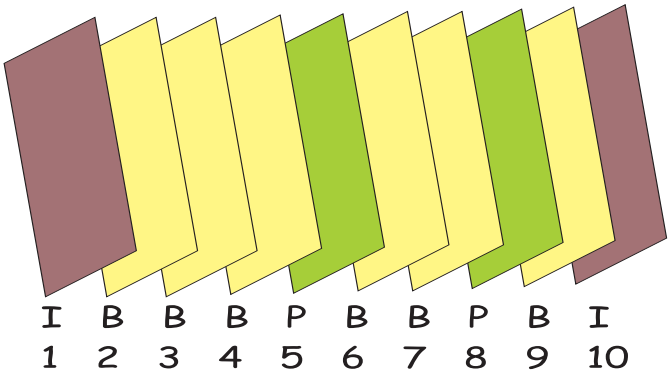

Figure 2.10: Frame Viewing Order

using motion prediction and (if no motion match is found) intra-coding. Frame P5 is based on I1, while P8 is based on P5. The B frames perform motion prediction on two directions. This means that frames B2, B3, and B4 are based on frames I1 and P5; frames B6 and B7 are based on frames P5 and P8; and frame B9 performs prediction on frames P8 and I10.

Because the decoder must perform prediction on future frames, all reference frames must be available to the decoder before any predictions can be calculated. The viewed information is not ordered in the optimum sequence for decoding. To decode the sequence at the optimum speed, the frames are ordered as seen in Figure 2.11.

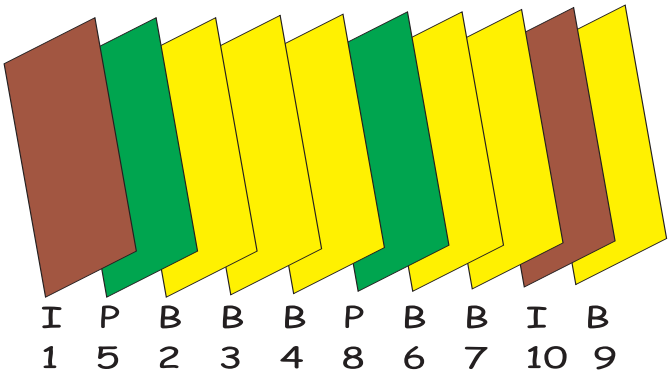

Figure 2.11: Frame Sent Order

Before any frame that requires any prediction is sent, all frames that are used as reference for the needed frames are sent. The first frame sent is I1. B2-4 need P5, so P5 is sent in the second

slot; B2-4 can now be sent. Next, B6-7 need to be decoded, but P8 is needed next. P8 performs prediction based on P5, which is already available. Once P8 is sent, B6-7 can be sent. This ordering algorithm is used for the whole video sequence.

An understanding of the framing structure provides a frame of reference to understand the process of capturing and encoding the video sequence.

Image Generation

To generate the image for MPEG compression, the video sequence must first be acquired digitally. The resulting sequence is made up of frames that are composed of pixels. An example format for these pixels may be Red-Green-Blue (RGB) format. To define any color, three reference vectors are needed. The format that is used in MPEG is YCrCb. Y is a luminance vector, and Cr and Cb are two color vectors [4]. The combination of these three vectors will render a full color image.

It is possible to under-sample the color matrices used to represent color in MPEG without significantly changing the perceived video quality. The formats used are 4:2:0, 4:2:2, and 4:4:4. These numbers represent macroblock layout, which is discussed in the Spatial Compression section. However, the sampling performed to acquire the image and convert it to MPEG block format is dependent on the desired macroblock layout. Figure 2.12 shows the sampling formats that are used for different macroblock distributions.

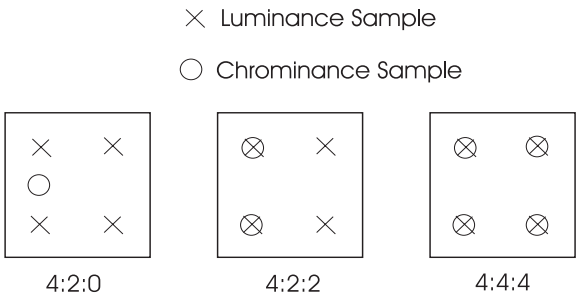

Figure 2.12: Image Sampling in Different Modes

Figure 2.12 shows the pixel pattern that is sampled. The complete sampling format is 4:4:4, where each pixel of the whole set is sampled once for every vector–every luminance pixel and each color pixel is sampled. The human eye is considerably more sensitive to luminance than color

information. It is possible to under-sample the color information in a video sequence, with little quality degradation. The two other formats listed in Figure 2.12, 4:2:2, and 4:2:0 are the two ways that MPEG under-samples color. The 4:2:0 format is designed such that for every square composed of 4 pixels, 4 luminance samples are taken, and each of the color vectors is sampled just once. The 4:2:2 format also samples every luminance pixel, but it samples every other color pixel.

The total number of samples will match the expected macroblock distribution. The sequencing of samples is dependent on the type of video (interlaced or progressive).

Once the video sequence has been captured, it is ready for compression.

Spatial Compression

The Y, Cr, and Cb vectors are assembled in eight-pixel by eight-pixel groups called blocks, the basic unit of video in the MPEG standard. Each block is converted from time data to frequency data by using a Discrete Cosine Transform (DCT), which is defined by

$$F(u,v) = \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\left(\frac{(2x+1)u\pi}{2N}\right) \cos\left(\frac{(2y+1)v\pi}{2N}\right) \tag{2.2}$$

where $u, v, x, y = 0, 1, 2, ...N-1$, x, y are coordinates in the 2-D matrix (sample domain), and

$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } u, v = 0 \\ 1 & \text{otherwise} \end{cases} . \tag{2.3}$$

The inverse DCT, which is needed in the decoder, is defined by

$$f(x,y) = \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} F(u,v) \cos\left(\frac{(2x+1)u\pi}{2N}\right) \cos\left(\frac{(2y+1)v\pi}{2N}\right). \tag{2.4}$$

Figure 2.13 shows the distribution of information on the DCT block. The higher frequency, high detail information is localized in the top left of the block.

Once the DCT operation is complete, the block is quantized. The level of quantization will determine the level of detail available to the decoder. The coarser the quantization, the higher the

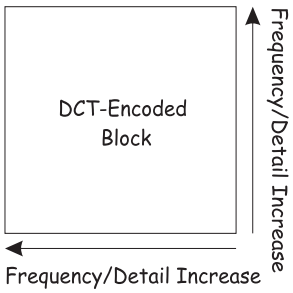

Figure 2.13: Detail Distribution in DCT-Encoded Block

probability that the high-frequency information, whose value is already fairly close to zero, is completely zeroed-out. The quantized block should have a large number of zeroes localized in the top left.

All blocks are assembled into slices, horizontal sets of blocks. The length of a slice is never longer than the width of the viewed image. Frame information is passed in slices. These slices do not, by definition, need to cover the whole viewing area. However, the MPEG standard does not provide guidelines for decoder action when slices are not covered. If slice information is missing when the sequence is encoded, it is up to the encoder to make sure that no information is sent that is referenced to any blocks within the slices. The decoder assumes that any area that is not covered by slices is not encoded. Furthermore, the decoder assumes that no data is referenced to the missing information. The standard does not predict the behavior of the decoder if these areas were to be used for referencing.

Information on blocks is further compressed by taking advantage of the image acquisition characteristics of the human eye. Since the eye is more susceptible to luminance than chrominance information, some chrominance information can be omitted from the transmission. The blocks are assembled into macroblocks. Figure 2.14 presents the different layouts that the macroblocks may have.

Figure 2.14 shows the way the information is assembled for transmission in a video stream. A macroblock is a composite structure covering 16-pixels-by-16-pixels on the screen. Each macroblock is composed of three types of blocks; a luminance, or one of either chrominance blocks (seen as different colors in Figure 2.14). Since the sampling technique used determines the amount of data

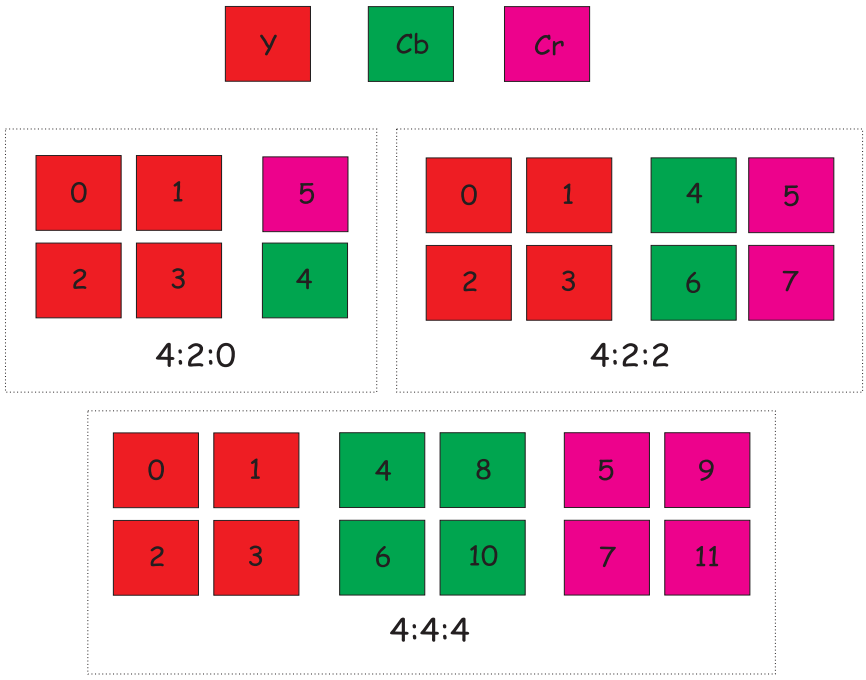

Figure 2.14: Macroblock Layout

available in each color, the arrangement of blocks within the macroblock has to be modified to accomodate different data sets. For example, if the 4:4:4 sampling method was used, then each macroblock is composed of four blocks of Y (luminance), four blocks of Cr (one of the chrominance vectors), and four blocks of Cb (the other chrominance vector), totalling twelve blocks needed in the transmission to compose a full macroblock. However, if 4:2:0 sampling were used, four Y blocks would still be available, but data equivalent to only one Cr and one Cb would be available. Therefore, only six blocks would be available for transmission. The 4:2:2 format is composed of four Y blocks, and data equivalent to two Cr and two Cb blocks, totalling eight blocks needed in the stream to compose a full macroblock. Macroblocks are composed sequentially in a slice. This means that if a slice across a frame is composed of ten macroblocks, and the format used is 4:2:0, then the slice header would be followed information concerning a total of sixty blocks, mapped in the sequence presented in Figure 2.14.

Temporal Compression

Although spatial compression yields a significant reduction in data transmitted, temporal compression takes advantage of the redundancy inherent in video sequences to further reduce the amount of data needed to describe a video sequence to an MPEG decoder.

There are two types of frames that perform motion prediction: the B and P frames. The P frames perform motion prediction based on the previous I or P frame. B frames, on the other hand, perform motion prediction based on the previous and next I and P frames.

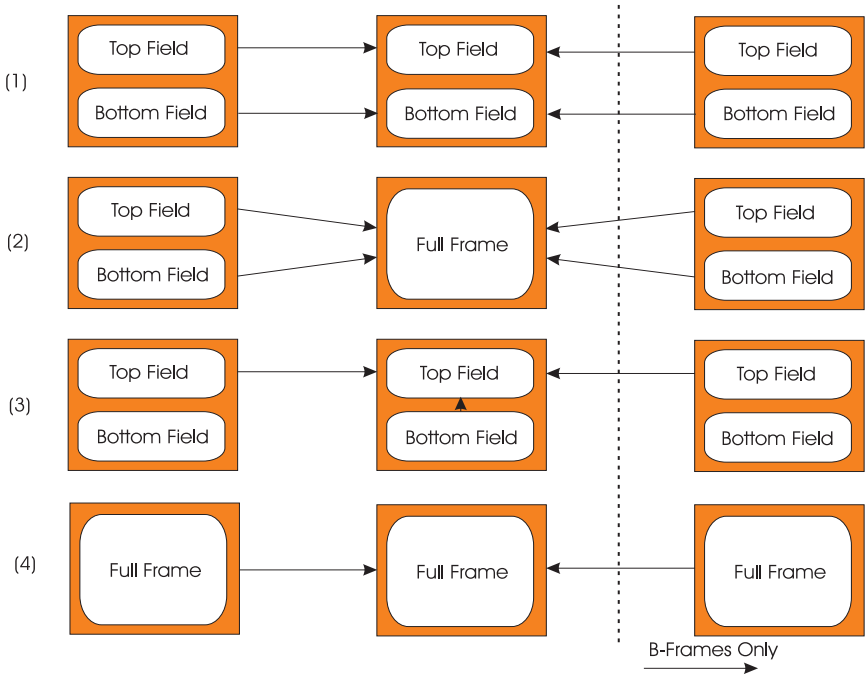

Figure 2.15: Types of Motion Prediction

Motion prediction is performed as seen in Figure 2.15. The frame is split into two pictures, the top field picture and the bottom field picture. The combination of both fields yields a whole frame. In Figure 2.15, part (1), each field is used to predict the corresponding field on the predicted frame. In the case of B frames, both the previous and next fields are taken into account. In part(2), the top and bottom fields are combined to predict the whole frame. In part (3), a combination of prediction is shown. Here, the prediction from the corresponding field on the previous and next reference frames are used as guides as well as the neighboring field. In part (4), a whole frame is

predicted by using another frame.

The unit of motion prediction is the motion vector, a differential value that maps coordinates of similar macroblocks from one frame to the next. To determine a motion vector, the encoder must first select a macroblock to encode. This macroblock is then compared to equal-sized areas in the reference frame at half-pixel distance increments. When a match is found, the vector is calculated and stored. If no match is found, the macroblock is intra-coded. If the frame being encoded is a B frame, the process is repeated with the other reference frame. Once the macroblock is encoded, the encoder moves on to the next macroblock. To decode the motion vector, the decoder reads the motion vector(s). If more than one is available, the two reference vectors are read and the average is calculated. The new macroblock is then placed in the calculated coordinates.

One point to note in the encoding of motion vectors, the encoding algorithm is considerably more complicated than the decoding process. Whereas the encoder has to search each entire frame to find a match to a particular macroblock, the decoder must only read a vector and place a macroblock. In typical communication systems, the decoding process is the most complicated algorithm to implement. For example, the complexity of a convolutional encoder pales when compared to the complexity of a Viterbi algorithm. In MPEG, on the other hand, the encoding process is the complicated part of the algorithm, making it relatively easy (and inexpensive) to implement a real-time MPEG decoder, such as the Direct-Broadcast Satellite box. On the other hand, real-time MPEG encoders are considerably more expensive and difficult to build.

2.2.3 Program Formats

The previous section, Video Coding, described how video coding is performed on a video clip. This video sequence is just a small part of the MPEG standard. To be displayed to the end-user, it must be transmitted in the context of a whole program, which contains information that allows the decoder to compile several elementary streams (sequences) into a single program. The single program contains video and audio information that is synchronized. Also, in the more advanced settings, the program allows the MPEG to transmit extra information that will provide other services to the end-user such as closed-captioning and premium services.

MPEG offers two formats for the transmission of programs: Program Stream and Transport Stream. Program Stream is a variable-length format that allows the transmission of a single program to the end-user. This format is designed for media that is relatively error-free since the loss of information would probably result in a serious synchronization problem. Another major limitation in this format is the lack of programming information, which could be used to arrange a collection of programs into a single, coherent service. Program Stream is the format that MPEG-1 uses.

MPEG-2 was designed with two goals in mind: the expansion of services and backward compatibility. The expansion of services is made possible by improved computer technology that allows the cost-effective decoding of complex MPEG streams. Backward compatibility allows new technology to merge with the pre-existing infrastructure. Because of the compatibility requirements of the standard, MPEG-2 supports the program stream format. To support a new suite of services, a new format for the delivery of MPEG programs, the Transport Stream (TS) format, was designed. TS format is a packet-based format that allows the multiplex of several programs onto a single program stream. Since the programs are packet-based, the identification and recovery of errors is greatly simplified. The packet format allows the use of complex error detection/recovery algorithms not available in a continuous stream. Error recovery is simplified because an error can be easily detected. Since they are easily detected, the decoder is well-armed to deal with problems by either requesting a re-transmission or using another technique to handle the error in the stream.

TS is only one of the DAVIC formats for data packets, as explained in the following section.

2.3 Data Packet Construction

The data format for the digital system described by the DAVIC guidelines is packet-based. The guidelines list three formats for data packets; two for the downlink and one for the uplink. The downlink packets are both 188 bytes long. One is MPEG-2/Transport Stream standard, and the other consists of ATM packets packed inside 188-byte-long packets. The uplink data format is ATM.

Since a detailed description at this point would not contribute significantly to the knowledge base needed to understand the issues discussed in this paper, a thorough description of the DAVIC and

ISO-13818-part-1 guidelines is contained in Appendix A.

The main piece of information that is derived from the description in Appendix A is the MPEG-2/TS packet length, 188 bytes.

The packet length used in the Transport Stream section of the MPEG standard is of consequence because it is the basis for determining the maximum mean burst event described in the analysis section of the paper.

2.4 Channel Model

Since this project is aimed at the study of an LMDS system, the LMDS channel (27-31 GHz) must be addressed. However, the simulation software, once written, should not be limited to LMDS only. Therefore, it is important to include other channel models to be able to study a wider variety of phenomenon. Other channels also have certain characteristics such as multipath fading that need to be explored.

2.4.1 LMDS

The LMDS channel occupies several bands. Table 2.6 describes the bands covered by each block auctioned by the FCC.

Table 2.6: LMDS Spectrum

Block	Band Bottom (GHz)	Band Top (GHz)	Width (MHz)
A	27.500	28.350	850
A	29.100	29.250	150
A	31.075	31.225	150
B	31.000	31.075	75
B	31.225	31.300	75

The auctioned wide-bands have no FCC-mandated use restrictions, except for radiated power limits. Since the spectrum is so wide, it can accommodate channels that are wide enough to permit

broadband services. The DAVIC guidelines call for channels that are anywhere between 2.5 MHz and 40 MHz wide.

This band is currently being used by satellite telecommunication systems (Ku-band), primarily for point-to-point communications using highly directive antennas.

Studies of the channel for terrestrial applications have focused on two areas, multipath and fading. The multipath studies have focused on the shadowing problem, and the fading studies have centered on vegetative fading and signal depolarization.

Multipath

The early study of the LMDS channel centered on the multipath characteristics, which were derived from field measurements. A US Department of Commerce study [5] disclosed findings concerning the multipath and diffraction characteristics of the LMDS channel. Measurements were performed with a 20 MHz wide channel centered at 30.3 GHz. The transmitter used a twenty-six degree wide horn antenna. The main tool used to study multipath elements was the delay spread of the signal. The receiver used a narrow beamwidth antenna (5.5 degrees). The delay spread of the signal was between 0.7 ns and 10 ns, with a median value of 1 ns. This data pointed to highly attenuated multipath elements in the transmission. In this measurement campaign, trees had no foliage, so vegetative effects due to foliage could not be studied.

A propagation measurement performed in Brighton Beach, New York, in 1995 [1], revealed similar results concerning the channel. In this study, the main effect studied was system coverage. The goal of the study was to determine the percentage area coverage of the signal. Table 2.7 from [1] shows the results from that measurement campaign.

The table presents the percentage of the area that was not covered by the 28 GHz signal. The area where the measurements were taken was an urban setting with three- or four- story buildings. This study supports [5] the assertion that there was little or no multipath in the LMDS environment. It is important to note that the only measurements that recorded a significant signal level were from samples taken at points with a line-of-sight (LOS) to the transmitter with little or no blockage.

Unpublished results of measurements performed at MPRG at the 38 GHz band have revealed

Table 2.7: LMDS Blockage [1]

Antenna Height	All Locations	< 3 km from transmitter	< 2 km from transmitter	< 1 km from transmitter
11.3 m	32%	32%	28%	14%
7.3 m	54%	55%	50%	29%
3.4 & 4.0 m	74%	73%	70%	52%

surprising multipath components in a point-to-point microwave link. During and directly after a rainstorm, water will tend to collect in pools on flat surfaces. Surfaces that previously were non-reflective suddenly have a flat, reflective surface, creating a multipath element. This multipath element does not change as a function of time, so it will not lead to a fading channel. However, this multipath element does change the delay spread of the incoming signal and should be taken into account when high-speed wireless links are being designed. The simulation and modeling presented in this paper assumes the delay spread is lower than the data rate.

Fading

Since it was determined that there is highly attenuated multipath in this environment and given that directive antennas are used on at least one side of the communications link (the user terminal), it is safe to assume that there is little or no coherent additive path cancellation, or Rayleigh fading, which occurs due to active cancellation of multipath elements offset by small time delays [6]. Although there are no references to the study of moving channel characteristics at the Ku-band, it is safe to assume that there is no fading due to multipath cancellation.

However, the channel does have some fading characteristics due to vegetation blocking the line-of-sight component of the communication link. [7] states that the effect of foliage on the link caused a fair amount of fading. This fading could be approximated by a Nakagami-Rice distribution with a high k factor (over twenty). The Rician fades were deeper than the fades measured, so a lognormal distribution could also approximate the effect of the foliage on the link. The speculated cause for the fading is leaf diffraction. This hypothesis was supported by a measured increase in signal

time variability in windy conditions. In other words, windy conditions caused the angle of attack of the leaves to change faster than in calm conditions, causing a larger change in the diffraction characteristics of the tree.

In rainy conditions, the signal path loss increased 10 dB over the expected loss due to rain in the path. This difference is probably due to water accumulation on the leaves, causing extra attenuation that would not be present if the tree were dry.

2.5 Video Quality Assessment

The assessment of video quality rests on the subjective analysis of the viewer. Whereas a typical communication link can be characterized by such quantifiable measures as BER and SNR, the quality of a video clip cannot be quantized as easily.

Several different methods can be used to assess the quality of video. One method is the quantification based on subjective viewer assessment. In other words, a group of people is selected, and these people watch the video and rate it on a fixed scale. This method has several drawbacks. First, a group of people willing to view a set of video clips is necessary. Getting a statistically significant group of people to sit through a series of video sequences is no easy task. Also, not everyone's scale is the same. For some people, a certain degradation may be acceptable, but for others the same level of image degradation may be completely unacceptable. To complicate matters, humans do not have a fixed quality scale. Whereas an image may seem completely unacceptable at the beginning of a series of clips, the same image may be regarded as having a far better quality after a whole series of similarly distorted images is seen [8].

A considerable amount of effort has been placed in establishing a metric for video quality. Some of these metrics are designed to measure the result of operations that subtly degrade image quality, such as the effects of compression algorithms on a raw video image [9] [10]. The primary aim of these metrics is to maximize capacity (by minimizing bit-rate) while maintaining a reasonable picture quality. Other metrics, such as the Square Root Integral (SQRI) are designed to look at image sharpness and the subjective effect of defocusing [11]. Some measures, such as the Discriminal Difference Diagram (DDD) look at horizontal variations in the image [11]. Another measure

integrates the Modulation Transfer Function across a desired frequency range yielding the MTFA [12]. These measures of quality were designed to observe particular aspects of video signals, such as edge detection, or they were designed to evaluate the quality of a video sequence without large artifacts, such as those seen in digital video sequences with bit errors. The simulation yields errors that are comparable to the loss of packets on an ATM network. These errors are very dramatic and entirely different from the effects seen from lossy compression. For example, lossy compression would yield a sequence that lacks definition or is blurry. On the other hand, a bit error would appear on the video clip as a corrupted block, a very large, square form of salt and pepper noise; in some cases it may appear as the loss of the large part of a slice.

One form of error assessment that can be used in video is the peak-signal-to-noise-ratio (PSNR), defined by Equation 2.5 where n is the number of bits per pixel [8].

$$PSNR = 10 \log_{10} \left(\frac{2^n - 1}{MSE} \right) \tag{2.5}$$

This equation yields the video clip's overall mean-square pixel error and is the metric used here. There are certain limitations to this approach to performance analysis. The PSNR was originally designed to evaluate images with slight degradation in quality and no large artifacts. This method does not evaluate well the effects of large artifacts on the human perception of an image [8]. Bit errors in a digital video sequence will generate large artifacts, which will yield an unacceptable video quality for broadcast applications. However, PSNR does provide a basic measure of relative video quality; it provides a measure of the relative number of artifacts that appear on the image area. PSNR is also a very popular method to provide an objective measure of quality in digital video research. Since the only form of distortion seen in this set of simulations is distortion resulting from bit errors in an MPEG video sequence, video quality in this thesis is defined as: the relative number of artifacts appearing in a sequence whose magnitude is measured by the PSNR.

Chapter 3

Simulation

In this chapter, we present RSVP, a flexible software tool that allows communication system designers to quickly and easily test different system layouts and coding techniques. A search of previous simulation efforts was performed; this revealed limitations in the effects modeled in simulations.

Typical video simulators implement a simplified communications system or channel model. Software found in the literature often present a reasonable channel model, but an overly-simplified communications system design [14]. In some cases, the channel may be fairly complicated, including a fading envelope [15]; however, these simulations lack the sophistication at the communications system level needed to simulate the desired system configuration parameters. Since digital video research has historically been the domain of computer science, the focus of research has often concentrated on the network issues, such as packet loss, and its effect on video quality. Simulations of video transmissions in packet networks have been implemented by several research groups [16][17]. Some simulations of network transmissions also simulate physical layer errors, such as bit-error. These bit-errors are generated by switching the state of a bit based on a single random test based on the system configuration's probability of bit-error, not taking into account any possible bit-error clustering [17].

RSVP was developed because no software tools were found that have the short execution time necessary to evaluate video sequences in a reasonable length of time, the flexibility needed to test the different configurations the project requires, and the complexity needed to accurately model a

communications system.

3.1 Simulation Capabilities

The tool that was created for this project is a flexible piece of software that allows the user to explore several aspects of wireless digital video communications. Details for the implementation of the simulation are available in Appendix B.

3.1.1 Communication System

The communication system is the core of the simulation. The architecture of the communication system simulator has a modular, flexible configuration. The organization of the simulation is such that the user is capable of changing the parameters defining a module or whether or not that module is active from using a GUI (Graphical User Interface).

The simulation was designed to allow the modeling of several different wireless environments. A Gaussian noise generator is complemented by Rayleigh-, Rician-, and LogNormal-fading envelopes. The simulation can also generate co-channel and adjancent-channel interference.

3.1.2 Video

The video simulator is split into two parts: a video viewer and a packetizing application.

The video viewer is capable of displaying MPEG-1 or MPEG-2 video elementary streams. Some robustness has been added to the viewer to allow it to display video sequences with very high BERs.

The packetizing application is an application independent of the video viewer. Its function is to prepare an elementary stream for transmission in a communications system. The formats that the application can currently encode and decode are: MPEG-2/Transport Stream, Asynchronous Transfer Mode (ATM), and Aggregated ATM Cell Encapsulation.

3.2 Simulation Verification

Several BER curves were run to test the performance of the communication system. This section will show the performance of the system using QPSK modulation, a Reed-Solomon code, and a convolutional code. The simulated performance is compared to the theoretical bounds. All simulations are performed in an AWGN channel.

3.2.1 Modulation

A QPSK tranmission in an AWGN channel can be described by Equation 3.1.

$$BER = Q\left(\sqrt{2\frac{E_b}{N_o}}\right) \tag{3.1}$$

where $\frac{E_b}{N_o}$ is a fraction (not in dB) and Q(x) is defined by Equation 3.2.

$$Q(x) = \frac{1}{2}erfc\left(\frac{x}{\sqrt{2}}\right) \tag{3.2}$$

The performance of the simulation was tested using two different filter roll-offs ($\alpha = 0.2$ and $\alpha = 0.35$). Since Nyquist filters are used, the performance of the system should be independent of the filter roll-off used.

Figure 3.1 shows the performance of the simulation of a QPSK-modulated system with nyquist pulse shaping with $\alpha = 0.20$ and $\alpha = 0.35$.

The curves for both filter roll-offs yield the same BER. This shows that, as expected, the relative performance of the systems will remain constant for different Nyquist filter roll-offs.

The simulated curves match the theoretical curve very closely. The theoretical curve describes the performance of a QPSK system in an AWGN channel with no ISI (inter-symbol-interference). The differences between the two curves are very small and can probably be accounted for by accumulated rounding and precision errors in the computer arithmetic.

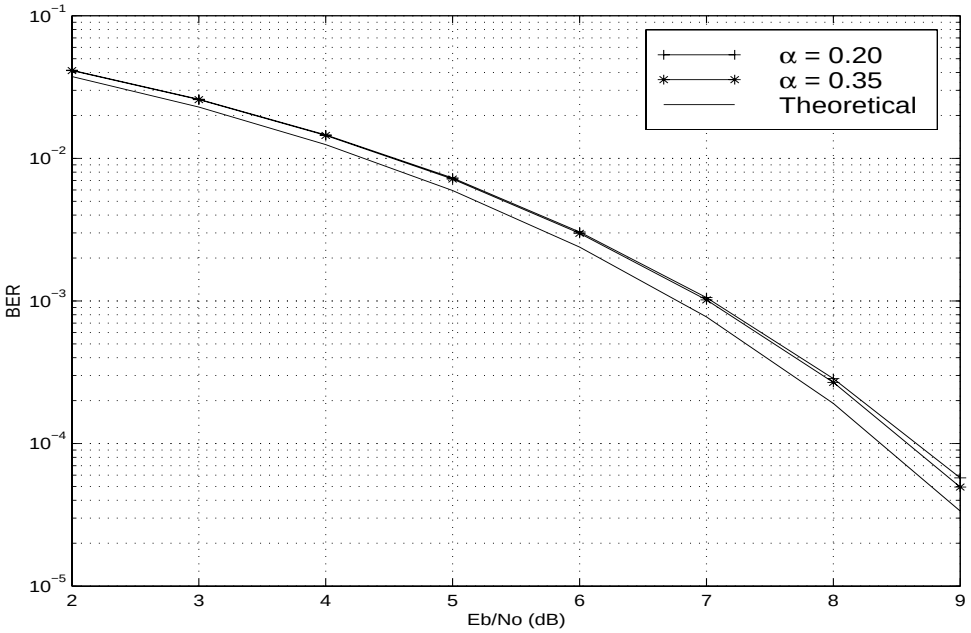

Figure 3.1: BER Curve, QPSK in AWGN Channel

3.2.2 Reed-Solomon

The Reed-Solomon code presented several challenges for validation. There is no analytical curve that will describe the exact performance of the code. However, there exists an approximation of the expected performance of the Reed-Solomon code [18]. Equation 3.3 gives an approximation for the probability of symbol error in a Reed-Solomon code.

$$P_0 \approx \frac{N+1}{2N^2}\left[D_{min}\sum_{i=t+1}^{D_{min}}\binom{N}{i}P_c^i\left(1-P_c\right)^{N-i}+\sum_{i=D_{min}+1}^{N} i\binom{N}{i}P_c^i\left(1-P_c\right)^{N-i}\right] \tag{3.3}$$

where N is the number of symbols in the protected word, D_{min} is the code's minimum distance, and P_c is the probability of symbol error. The code's minimum distance is defined by Equation 3.4.

$$D_{min} = 2t + 1 \tag{3.4}$$

t is the code's error correction capability. P_c is defined as the probability of symbol error. Each

symbol of a Reed-Solomon code word is composed of at least one bit. Since the channel used is Gaussian (memoryless) and since there is no bit interlaving performed in the trial, it can be assumed that the bit errors are independent. Since the bits are independent, Equation 3.5 describes the probability of symbol error (where the symbol is composed of contiguous bits).

$$P_c = 1 - (1 - p)^n \tag{3.5}$$

where p is the probability of channel bit error and n is the number of bits per symbol. Since Equation 3.3 returns the probability of symbol error in a corrected word, the inverse of Equation 3.5, seen in Equation 3.6, can be used to recover from Equation 3.3 the probability of bit error in the corrected stream.

$$p_0 = 1 - \sqrt[n]{1 - P_0} \tag{3.6}$$

Figure 3.2 shows the performance of a (255,239) Reed-Solomon code in a QPSK-modulated transmission in an AWGN channel.

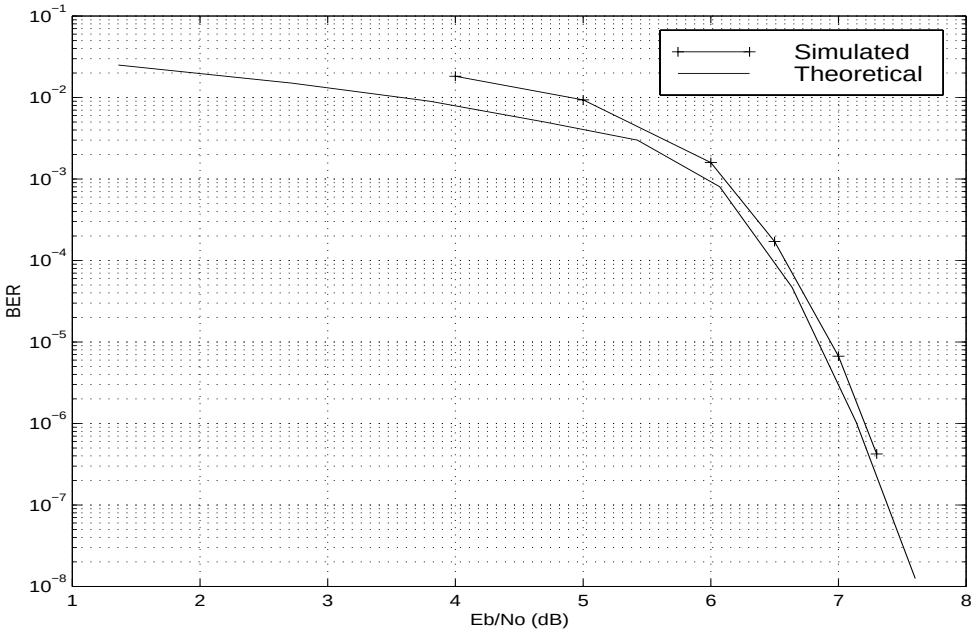

Figure 3.2: BER Curve, Reed-Solomon in AWGN Channel

The simulation results show that the simulation performance is worse than the theoretical curve at low $\frac{E_b}{N_o}$ values, but converges with the theoretical curves at higher $\frac{E_b}{N_o}$. The difference occurs because the theoretical approximation uses the D_{min} of the block code to determine its performance. At high $\frac{E_b}{N_o}$ values, the D_{min} value is the dominant determining factor in the error performance. However, as $\frac{E_b}{N_o}$ decreases, the D_{min}'s role in determining the failure of the code decreases considerably. At these lower values, the behavior of the code is affected by other factors, which are not modeled in the theoretical curve. The absence of these factors in the theoretical curve predicts a better performance than the code really yields.

3.2.3 Convolutional Code

The convolutional code is one of the most complicated modules that is implemented in C. The testing of the convolutional code was done in two parts. First, the module was tested at its base rate ($r = \frac{1}{2}$) using both soft and hard decision decoding. Once the basic module was validated, the module puncturing was tested.

Base Rate

The convolutional code was validated by using a bound on the code's performance. The calculated bound is based on the first-error-event probability of the Viterbi decoder. Since the code is linear, a simple transmission case is used (all-zeros transmission). The probability of the decoder path diverging and converging with the true path (first-error-event) is calculated. The probability of bit error is based on the first-error-event probability [19].

A convolutional decoder fails to decode a word properly when the least-distance path diverges from the true data path for a certain number of states until it converges again with the true path. Each of these divergences will have a certain error distance (D^d), and it will incur a certain number of data bit errors ($N^{f(d)}$). Each of these path will occur a certain number of time (a). A polynomial of the form described in Equation 3.7 can be contructed to describe these diverging paths.

$$T(a, D, N) = \sum_{d=d_{free}}^{\infty} a_d N^{f(d)} D^d \tag{3.7}$$

Since both hard and soft decision perform decoding on the same polynomial, the probability of bit error (P_b) for both cases is based on $T(a, D, N)$. $P_{b(soft)}$ for soft decision is bounded by Equation 3.8.

$$P_{b(soft)} < \left. \frac{dT(a, D, N)}{dN} \right|_{N=1, D=e^{-\frac{RE_b}{No}}} \tag{3.8}$$

where R is the rate of the code used. $T(a, D, N)$ in Equation 3.8 can be replaced by Equation 3.7, yielding Equation 3.9.

$$P_{b(soft)} < \sum_{d=d_{free}}^{\infty} a_d f(d) e^{-d\frac{RE_b}{No}} \tag{3.9}$$

This bound will tighten as the number of paths is increased. A recursive algorithm was built that explored the first few thousand paths that diverge and converge from the all-zero path in the Viterbi decoder, which yielded a polynomial of order fifteen. Figure 3.3 presents the simulated $r = \frac{1}{2}$ soft-decision simulation with the theoretical bound.

P_b for the hard-decision case is bound by Equation 3.10.

$$P_{b(hard)} < \left. \frac{dT(a, D, N)}{dN} \right|_{N=1, D=2\sqrt{p(1-p)}} \tag{3.10}$$

where p is the probability of channel bit error. Replacing 3.7 for $T(a, D, N)$ in 3.10 yields 3.11.

$$P_{b(hard)} < \sum_{d=d_{free}}^{\infty} a_d f(d) \left(2\sqrt{p(1-p)} \right)^d \tag{3.11}$$

Using the polynomial calculated in the recursive trellis search program and applying Equation 3.11 yielded a theoretical performance bound. This bound was compared to the simulated hard-decision, Viterbi-decoded convolutional code on Figure 3.4.

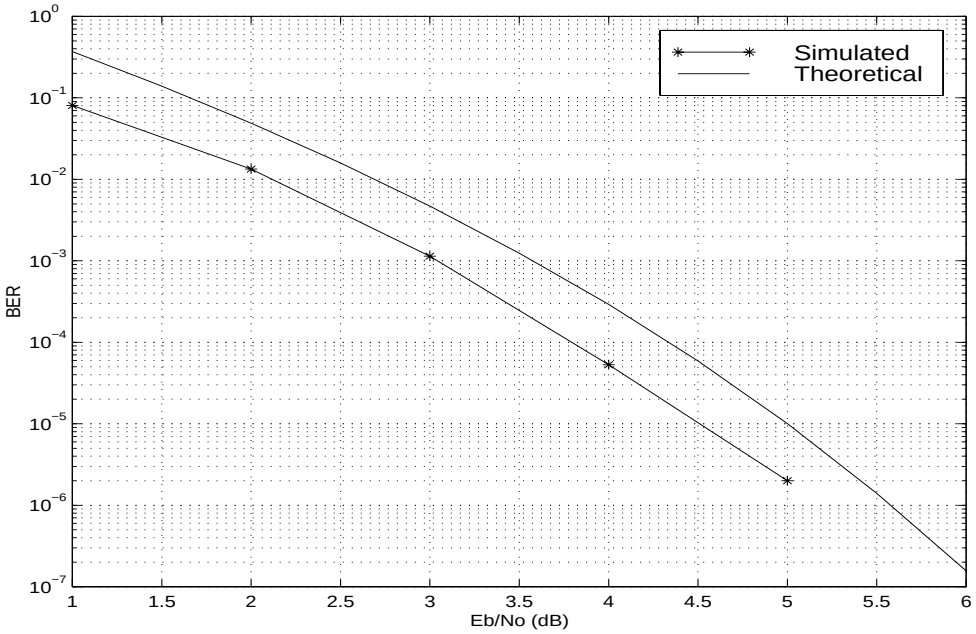

Figure 3.3: BER Curve, Soft-Decision Convolutional Coded in AWGN Channel

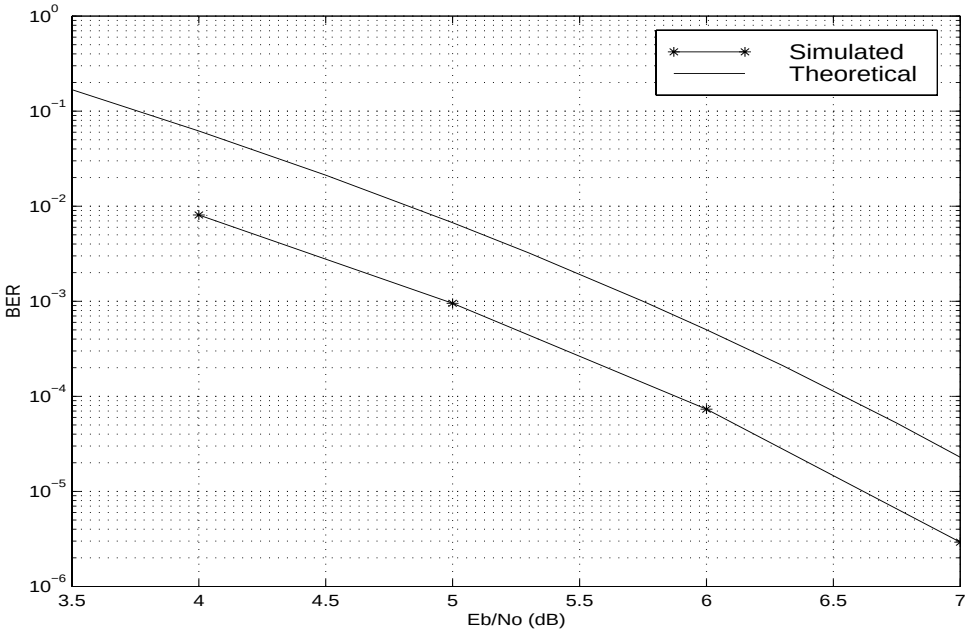

Figure 3.4: BER Curve, Hard-Decision Convolutional Coded in AWGN Channel

As seen in Figures 3.3 and 3.4, the simulation performed within the theoretical performance bounds. Once the validity of the basic convolutional encoder/decoder set was proven, the punctured version of the code needed to be validated.

Punctured

Since both soft and hard decision modes for the convolutional decoder were validated, the puncturing was tested using only hard decision. The main reason for selecting hard decision for the bulk of the testing is that system performance is considerably worse for hard decision decoding than it is for soft decision decoding (over 2 dB). The higher BER returned by the hard decision code means that shorted simulation trials are needed to generate a BER curve. Also, since the simulation of video sequences is the ultimate goal of the simulation, a higher concentration of bit errors means that a relatively short video sequence is needed to achieve a meaningful number of visible errors.

The method for validating the punctured hard-decision convolutional coding is very similar to the method used for the hard-decision full convolutional code. The recursive algorithm used to determine the error path polynomial defined in Equation 3.7 was modified to take into account the puncturing taking place in the decoder. Once the free distances were calculated, Equation 3.11 was used to determine P_b. The resulting theoretical curve was matched to the simulated system performance in Figure 3.5.

The performance of the simulation is bounded by the theoretical limit.

Once the modules in the simulation have been tested, the analysis of video transmission in a wireless system can be investigated.

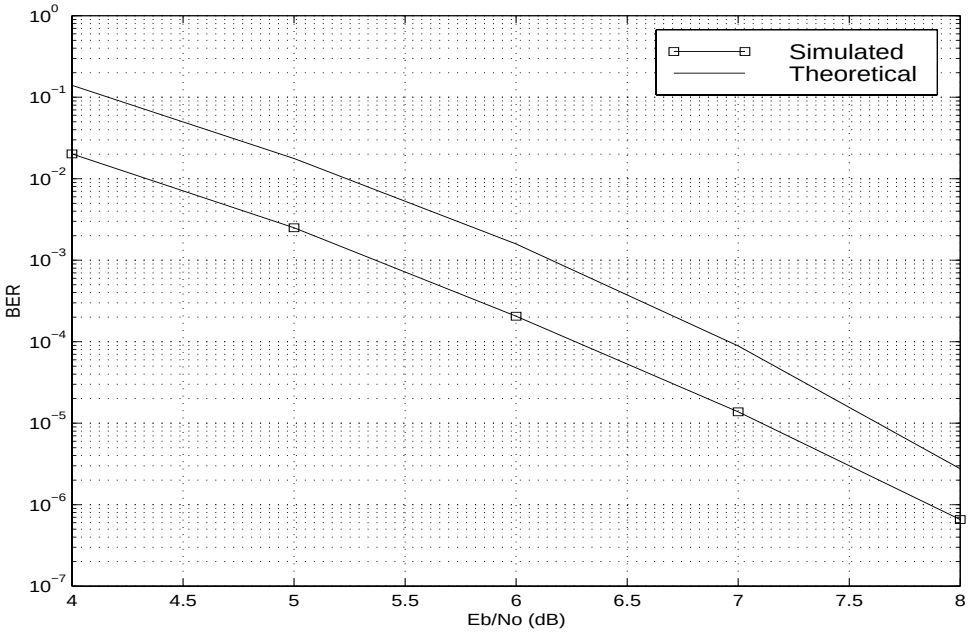

Figure 3.5: BER Curve, Hard-Decision Convolutional Coded in AWGN Channel

Chapter 4

Digital Video in a Noisy Channel

Testing focuses on two primary goals: assessing the effects of both error correction and co-channel interference on a video transmission and generating supporting evidence for the use of $\lambda_{distance}$ as a factor to predict video quality. The effects of different error correction algorithms on the video transmission are tested in an AWGN channel. The system is also tested when there is both Gaussian noise and non-Gaussian noise in the channel. The non-Gaussian noise is modeled through the use of an attenuated video sequence in the channel.

Another important product of these tests is the generation of data used to support the creation of a channel parameter that can be used to predict video quality. The parameter generated is $\lambda_{distance}$, the error-event-mean-arrival-rate. This parameter is discussed in detail in Section 4.1.3.

A path can be derived from the channel performance to the expected system video quality. Since the BER of a system can be determined either through a training sequence or based on the system's $\frac{E_b}{N_o}$ value, and since a system's λ_{error} is a function of system design and it does not change as a function of the environment, $\lambda_{distance}$ for the system can be calculated. Simulation data shows that a value for the expected video quality can be derived based on $\lambda_{distance}$.

4.1 Test Parameters

4.1.1 System Test Parameters

Since the goal of the project is to study video, a BER window had to be determined. To present a large enough number of errors to see distortions on the transmission, the video sequences will be tested above $BER = 10^{-6}$. This relatively high BER will assure that a fairly large number of errors will corrupt the video sequence. Also, since video is a relatively fragile data format, if the number of errors in the transmission is too high, it will be corrupted beyond the ability of the video decoder to decode the transmission. To keep the video transmission decodable, the BER of the testing is maintained well below 10^{-3}.

4.1.2 Test Sequences

Three video sequences were selected for testing: flowerseq.mpg (Figure 4.1), pongseq.mpg (Figure 4.2), and susiseq.mpg (Figure 4.3). All three sequences are of similar length (around 10 Mbit) and all operate at the same bit-rate (1.5 Mbps). All three sequences are MPEG-2, but they are also MPEG-1 compliant, which means that none of the files contain any extended field information.

Figure 4.1: 'flower' Test Video Sequence

Figure 4.2: 'pong' Test Video Sequence

Figure 4.3: 'susi' Test Video Sequence

The selected video streams have different characteristics. The 'flower' stream presents the panning over a vista; the vista changes little over the period seen in the video sequence. In such a sequence, the motion vector plays a key role in data-compression. The 'susi' stream is a portrait with several features that move and are hidden (such as hair moving from visible to behind the head), presenting several new information blocks in front of a static background. The 'pong' sequence shows a mix of the characteristics seen in the 'flower' and 'susi' sequences; the sequence pans over an area, presenting a fair amount of motion vectors combined with some action seen in the player's hidden features, such as arms, moving into and out of view. These sequences were selected in an effort to have a combination of features while maintaining a minimum amount of data to be simulated.

Each sequence was run three times at each power level and system setting. A total of nine trials (three trials of each sequence) were run per plotted point. The metrics by which the test cases are to be compared is given below.

4.1.3 Testing Metrics

The Peak-Signal-to-Noise-Ratio (PSNR), discussed in section 2.5 is a relatively simple method of determining video quality. This metric is used as the basic metric of video quality. Two other metrics, the BER and $\frac{E_b}{N_o}$ of the link are used to provide a point of reference to compare signal strength. The BER provides a simple guideline of system performance, while $\frac{E_b}{N_o}$ provides a power reference.

Communications systems are engineered based on the expected number of bit errors in a transmission (BER), a metric that may not be sufficient to characterize the performance of a digital communication system. It is the belief of the author that another set of metrics is necessary to complement the BER to predict the performance of a digital video system. A method of determining the "burstiness" of a channel is necessary to complement the BER metric to predict the quality of a video transmission.

Figure 4.4 shows the error distribution in a single transmission. The x-axis of the graph represents all n bit-errors, and the y-axis represents the distance (in number of bits) between the begining of the file and that error event. The y-axis of the graph is an absolute measure of distance in the

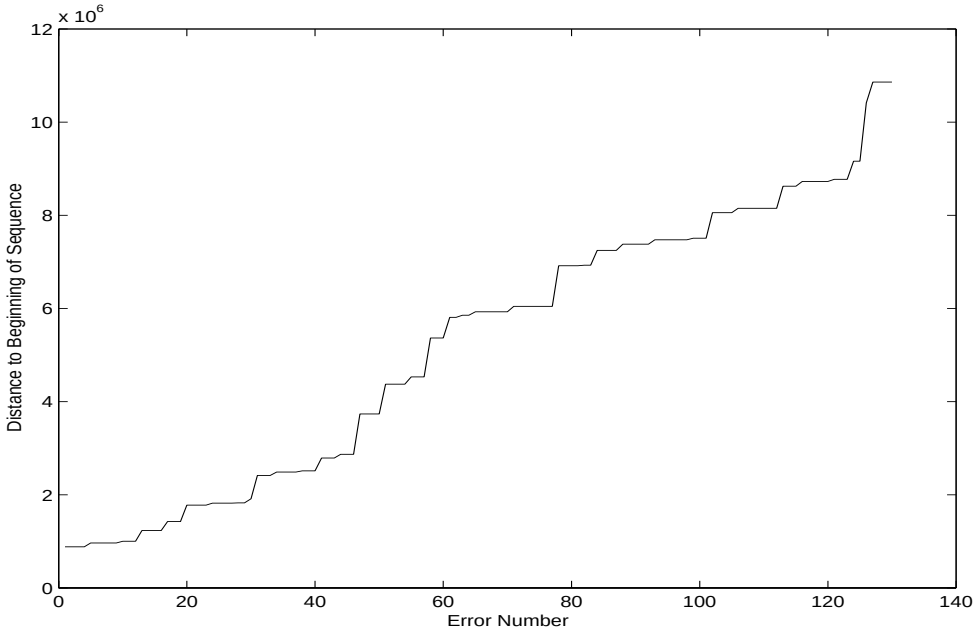

Figure 4.4: Bit Error Distribution over Time

transmitted stream. In other words, the stream contains roughly 11×10^6 bits. The x-axis is a sequential list of all bit-errors in the stream; there are around 130 errors in the stream. 130 errors distributed over a length of 11×10^6 bits means that the BER is roughly 1.18×10^{-5}.

Figure 4.4 consists of a series of horizontal lines, each horizontal line being comprised of several bit-errors occurring in a (relatively) short period of time. Each bit-error burst can be considered a single event of length τ_i, where i is the event number; there are approximately thirty-three error events on this graph. Each error-event is made up of a number of bit-errors less than n. If each bit-error burst is considered a single event, and if each event is associated with a length τ_i, then each event can be modeled as a random process of the form seen in Equation 4.1 [20], with λ defined as the *mean arrival rate*. Equation 4.1 is the pdf of the event interarrival time, or the separation between the beginning of each event, whose probability mass function is described in Equation 4.2, also known as a Poisson random process.

$$f_\tau(t) = \lambda e^{-\lambda t} u(t) \tag{4.1}$$

$$P_N(n,t) = \frac{(\lambda t)^n}{n!} e^{-\lambda t} u(t) \tag{4.2}$$

In Equation 4.2, n is the event number.

In the specific case of determining the mean bit-error burst length, λ is replaced with λ_{error}, the mean burst length. The mean separation between bursts, or the error-event-mean-arrival-rate $\lambda_{distance}$, is similar to λ_{error}. $\lambda_{distance}$ records the mean number of bits separating each error event. In other words, if Figure 4.4 were seen as a staircase, λ_{error} is the mean step width, and $\lambda_{distance}$ is the mean step height.

To fit the error event list seen in Figure 4.4 into a Poisson random process, the data has to be modified. An algorithm was written to search through the error list and assemble bursts into single events. The algorithm searches a sequence for bursts and parses it into individual events. A single burst is considered to be any number of bit errors within a packet length (1504 bits on an MPEG-2 Transport Stream packet). Two arrays of events are returned, one for the number of bit errors in each error event and the other for the number of bits between each error, whose mean is λ_{error} and $\lambda_{distance}$ respectively.

4.2 Gaussian Reference

Figure 4.5 shows the bit error distribution of a QPSK-modulated transmission in an AWGN channel with no error correction. The graph is composed of essentially a single diagonal line.

Since the channel has no memory and since the symbols are independent (no Inter-Symbol Interference), the probability of error for any one bit is independent of all others. Since the system environment does not change during the course of the simulation, the probability of bit error is the same for all bits. Therefore, one would expect that there would be errors in the receiver at fairly even intervals.

Tables 4.1, 4.2, and 4.3 describe the behavior of the system in a Gaussian noise environment with no error correction for the sequences 'susi,' 'pong,' and 'flower' respectively. These tables are summarized in Table 4.4. Differences in content lead to different concentrations of motion

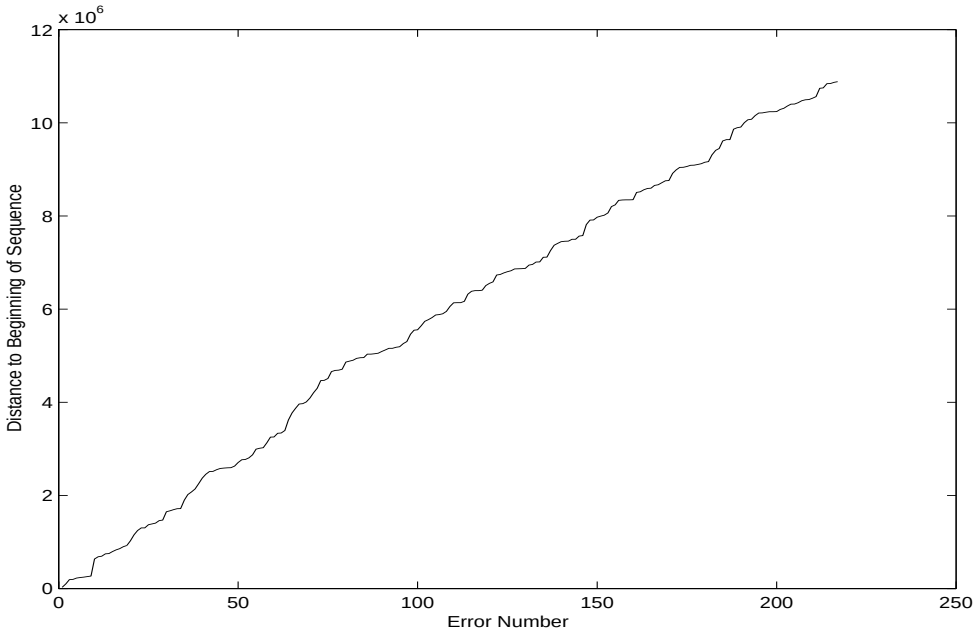

Figure 4.5: Bit Error Distribution, No Error Protection

Table 4.1: Mean Values for "Susi" with No Error Correction

$\frac{E_b}{N_o}$	λ_{error}	$\lambda_{distance}$	BER	PSNR
8.0	1.57	4.87×10^3	2.98×10^{-4}	10.6
8.5	1.23	8.77×10^3	1.37×10^{-4}	13.2
9.0	1.10	1.85×10^4	5.89×10^{-5}	16.4
9.8	1.02	7.90×10^4	1.29×10^{-5}	23.3

Table 4.2: Mean Values for "Pong" with No Error Correction

$\frac{E_b}{N_o}$	λ_{error}	$\lambda_{distance}$	BER	PSNR
8.0	1.56	4.87×10^3	2.96×10^{-4}	9
8.5	1.23	8.76×10^3	1.37×10^{-4}	10.8
9.0	1.08	1.85×10^4	5.81×10^{-5}	14.3
9.8	1.02	8.26×10^4	1.22×10^{-5}	20

Table 4.3: Mean Values for "Flower" with No Error Correction

$\frac{E_b}{N_o}$	λ_{error}	$\lambda_{distance}$	BER	PSNR
8.0	1.57	4.91×10^3	2.96×10^{-4}	9.7
8.5	1.22	8.66×10^3	1.39×10^{-4}	11.4
9.0	1.09	1.79×10^4	6.05×10^{-5}	14.8
9.8	1.03	8.19×10^4	1.24×10^{-5}	21.4

Table 4.4: Overall Mean Values with No Error Correction

$\frac{E_b}{N_o}$	λ_{error}	$\lambda_{distance}$	BER	PSNR
8.0	1.57	4.88×10^3	2.97×10^{-4}	9.74
8.5	1.34	7.48×10^3	1.90×10^{-4}	11.2
9.0	1.09	1.83×10^4	5.92×10^{-5}	15.2
9.8	1.02	8.12×10^4	1.25×10^{-5}	21.6

vectors and compressibility of DCTs; in the hopes of achieving a representative measure of overall performance, these three sequences were tested, and the results from these different sequences (of similar length, size, and coding format, but different content) were compiled into a single result. Table 4.4 is fairly representative of the sequences as a whole.

The amount of data that the presented tables summarize is very large. Each sample point (a single trial at a particular $\frac{E_b}{N_0}$) is the result of the analysis of a six- or seven-second sequence running at around 1.5 Mbps, resulting in a total length of roughly 10 Mbits. When error-correction coding is added to the sequence, the sequence length increases in length to as long as 21.7 Mbit. Ten samples are taken of each pulse-shaped bit, bringing the total sequence length to around 217 Msamples. Each trial is run three times per sequence (there are three sequences), so each trial at each point is run nine times, bringing the total number of samples analyzed to 1.95 Gsamples. Each test point in the overall results table represents the analysis of anywhere between 900 Msamples and 1.95 Gsamples. To minimize confusion created by the large amount of data generated by simulation, this data is then condensed into four numbers, the BER, λ_{error}, $\lambda_{distance}$, and PSNR.

There are several trends seen in Table 4.4 that should be looked at in some detail. First, λ_{error}

changes slightly as a function of BER. Even though the system architecture does not change, the mean burst length does change. The reason for this change is that the concentration of bit-errors increases. If the concentration is such that, even though the errors are evenly distributed, there is more than one error-event per packet, then the event-detection algorithm will identify these error-events as a single error-event. As the BER decreases, the concentration of errors throughout the sequence decreases, bringing the mean distance between errors to less than a packet length. For example, the convolutional error-correction code fails when the estimated path diverges from the true data path. This divergence may happen more than once per packet. Each divergence should be considered as a single error-event. However, because the error-event detection algorithm uses a single packet length as the basic unit when deciding if an error-event occurred, several divergences in the Viterbi algorithm will be evaluated as a single error-event. Even though λ_{error} changes as a function of BER, the change is not dramatic.

Another trend points to the fact that the BER can be derived from both λs. Equation 4.3 presents the relationship between the BER and λ_{error} and $\lambda_{distance}$.

$$BER = \frac{\lambda_{error}}{\lambda_{distance}}, \text{for } n \text{ large.} \tag{4.3}$$

A short proof of Equation 4.3 follows.

If a bit sequence with bursts of errors is parsed into individual bursts, two vectors can be created; $\vec{e}$ contains the number of bit errors in each burst, and $\vec{d}$ contains the number of bits separating the begining of each burst. Each vector is length n, the number of bursts in the bit sequence. Figure 4.6 contains a layout of bit stream and the parsing that has been done.

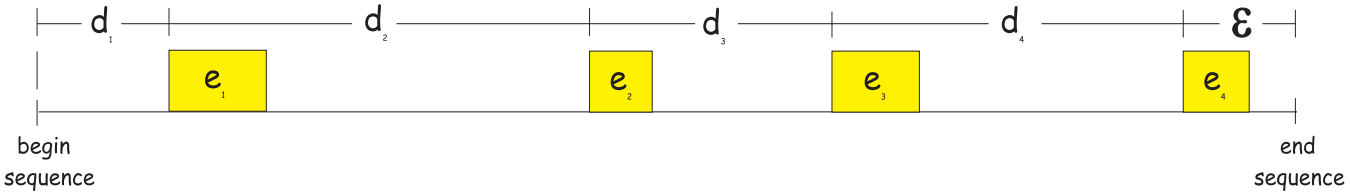

Figure 4.6: Bit-Error Burst Parsing Structure

The number of bit-errors can be calculated by adding the bit-errors listed for each burst as seen in

Equation 4.4.

$$N_{errors} = \sum_{i=0}^{n} e_i \tag{4.4}$$

As seen in Equation 4.5, the total number of bits in the whole sequence is the sum of the distance between all bursts plus the distance between the begining of the last error burst and the end of the sequence, represented by the number ϵ.

$$N_{bits} = \left(\sum_{i=0}^{n} d_i \right) + \epsilon \tag{4.5}$$

Since BER is the ratio of bit-errors to total bit count, Equation 4.6 can be used as a definition of BER.

$$BER = \frac{\sum_{i=0}^{n} e_i}{(\sum_{i=0}^{n} d_i) + \epsilon} \tag{4.6}$$

The variable ϵ is a function of the arrival rate of new bursts; it is not a function of the length of the sequence but of the system being evaluated. Thus, for large n, it is safe to assume Equation 4.7.

$$\epsilon \ll \sum_{i=0}^{n} d_i \tag{4.7}$$

Taking the assumption in Equation 4.7 into account, the limit of BER is defined by Equation 4.8.

$$\lim_{n \to \infty} \frac{\sum_{i=0}^{n} e_i}{(\sum_{i=0}^{n} d_i) + \epsilon} = \frac{\sum_{i=0}^{n} e_i}{\sum_{i=0}^{n} d_i} \tag{4.8}$$

Equations 4.9 and 4.10 define λ_{error} and $\lambda_{distance}$, respectively.

$$\lambda_{error} = \frac{1}{n} \sum_{i=0}^{n} e_i \tag{4.9}$$

$$\lambda_{distance} = \frac{1}{n} \sum_{i=0}^{n} d_i \tag{4.10}$$

The ratio of λ_{error} to $\lambda_{distance}$, as seen in Equation 4.11, is the same as Equation 4.8; therefore, Equation 4.3 is valid.

$$\frac{\lambda_{error}}{\lambda_{distance}} = \frac{\sum_{i=0}^{n} e_i}{\sum_{i=0}^{n} d_i} \tag{4.11}$$

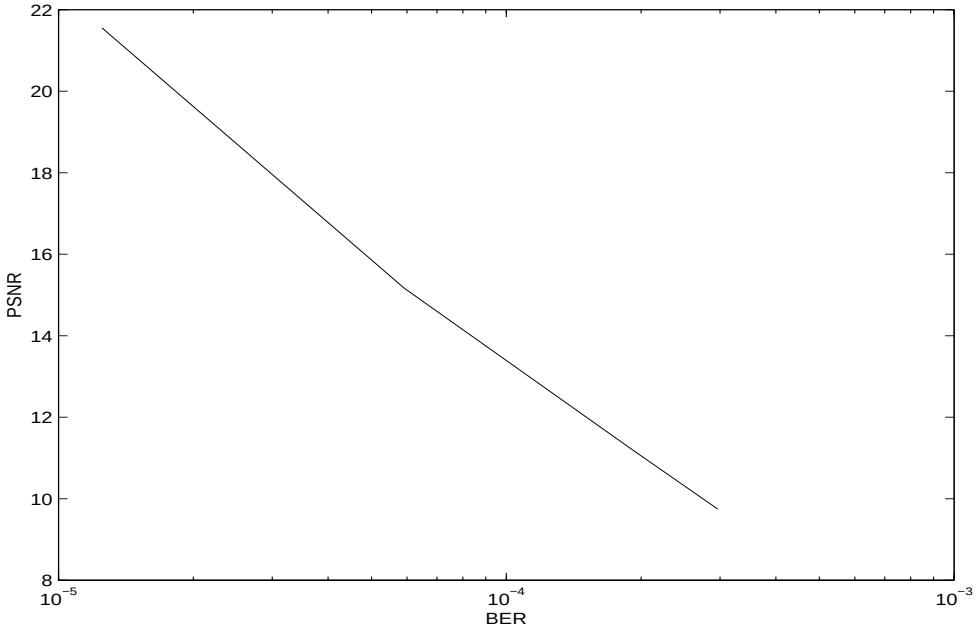

Figure 4.7: BER vs. PSNR for Sequence with no Error Correction

Finally, the PSNR seems to be a fairly linear function of the BER as seen in Figure 4.7. This relationship is intuitive since a higher number of bit-errors would lead to a higher number of visible defects in the video sequence. This relationship is seen only over the region in which the sequence is tested, and this relationship may not hold over other regions. This relationship will collapse as the BER increases. A video sequence below 10 dB PSNR has a very high risk of total decoding failure. Given the length of the sequence (only 10 Mbit), simulations with BERs too far below 10^{-5} do not yield credible results since the number of bit-errors achieved are not statistically significant.

4.3 Convolutional Coding

The convolutional code, described in 3.2.3, is a code that fails when an incorrect data path is selected in the Viterbi decoder. These deviations from the true data path occur in bursts of variable length. The probability of a burst of errors of a certain length is roughly proportional to the length of the burst. This relationship is due to the distance properties of the code. A shorter deviation is typically associated to a small free distance, and longer path deviations are associated with longer distances. This generalization gives an intuitive idea of the relationship of the code to the error distribution.

4.3.1 Bit Error Distribution

The BER curve for a $r = \frac{1}{2}$ convolutional encoder hard-decision Viterbi decoder system is seen in Figure 3.4. When $\frac{E_b}{N_o} = 6.5dB$, the simulation yields $BER = 1.2 \times 10^{-5}$, and the bit error distribution is seen in Figure 4.8.

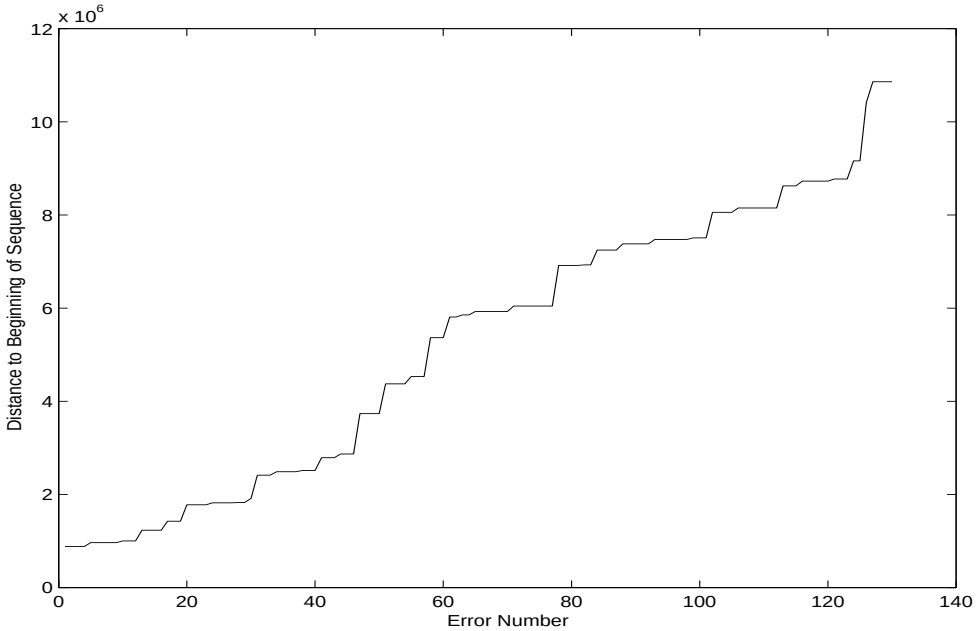

Figure 4.8: Bit Error Distribution, $r = \frac{1}{2}$ Hard Decision Convolutional Code

Zooming in on the bit-error distribution of the convolutional code shows that the errors are, for

the most part, clustered in relatively short bursts. Figure 4.9 presents a plot of the first fifty errors in the sequence.

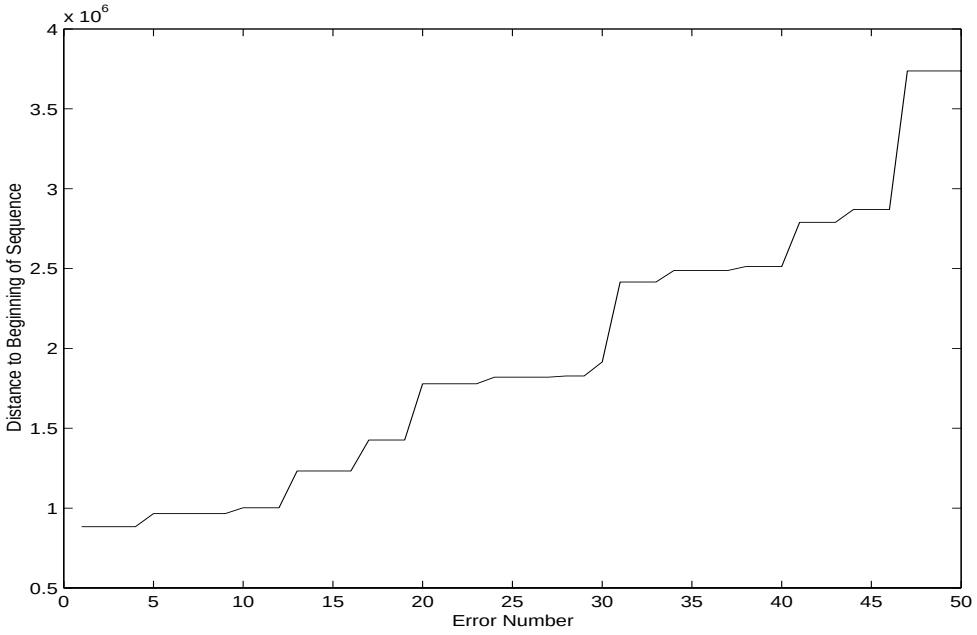

Figure 4.9: Bit Error Distribution, First Fifty Errors, $r = \frac{1}{2}$ Hard Decision Convolutional Code

To provide a point of reference for Figures 4.8 and 4.9, $\lambda_{error} = 3.61$ and $\lambda_{distance} = 3.02 \times 10^5$ for this particular scenario.

Several trials were run to determine the behavior of the convolutional code. Table 4.5 presents the results of these trials.

Table 4.5: Mean Behavior of Convolutional Code for all Sequences

$\frac{E_b}{N_o}$	λ_{error}	$\lambda_{distance}$	BER	PSNR
5.7	4.84	2.50×10^4	1.52×10^{-4}	16.2
5.9	4.61	5.14×10^4	8.96×10^{-5}	18.3
6.1	4.35	8.55×10^4	5.09×10^{-5}	20.3
6.3	4.11	1.54×10^5	2.69×10^{-5}	22.4
6.5	3.98	2.65×10^5	1.50×10^{-5}	24.7
6.7	4.02	4.73×10^5	8.49×10^{-6}	27.6

In Table 4.5, λ_{error} increases as the BER increases. There are two possible explanations for this behavior. The first possible explanation is that the number of deviations from the true data path in the trellis decoder increases. Since the burst algorithm looks for error events on the basis of a single packet-length, several small bursts will be interpreted as a large error burst. Another possible reason for this behavior, the added noise causes longer deviations from the true data path, increasing the mean burst length.

Similarly the no-protection simulations, the PSNR seems to be linear with respect to the BER, as seen in Figure 4.10.

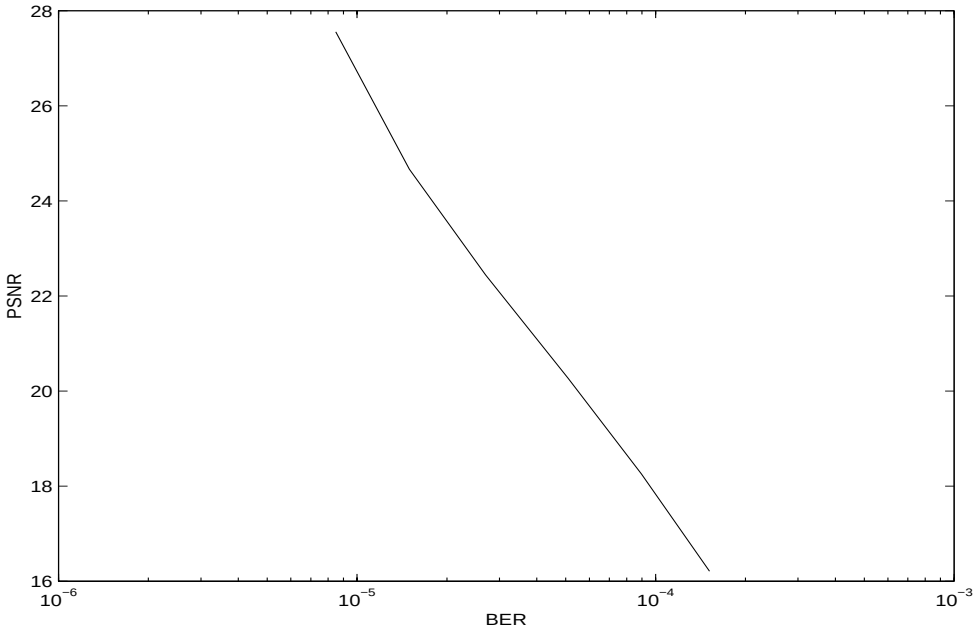

Figure 4.10: BER vs. PSNR for Sequence with Convolutional-Code Protection

4.4 Reed-Solomon Code

The Reed-Solomon (RS) code, described in 3.2.2, is a block code that behaves in a binary way; it will either perfectly decode a word (packet), or it will completely fail. This instance of the code (255,239) will correct any word that has as many as eight symbol errors, which may be anywhere between eight and sixty-four bit errors for the eight-bit symbol case. A set of trials, an example of

which is seen in Figure 4.11, were run, and several statistical metrics concerning these trials were collected.

4.4.1 Bit Error Distribution

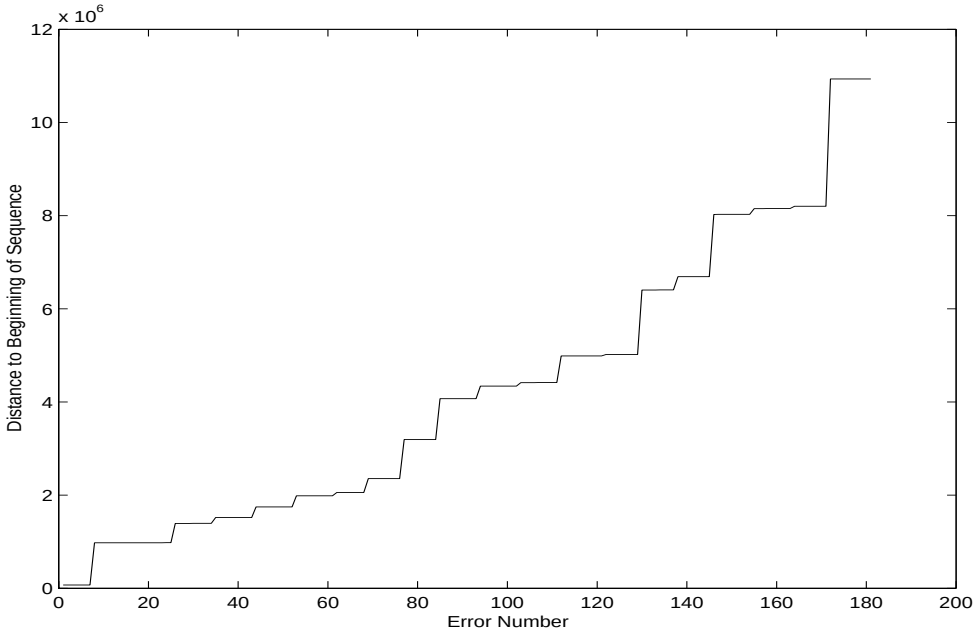

Figure 4.11: Bit Error Distribution, Reed-Solomon (255,239) Code

A zoomed version of Figure 4.11 is seen in Figure 4.12.

The shown sequence was simulated with $\frac{E_b}{N_o} = 6.9dB$, and a resulting $BER = 1.65 \times 10^{-5}$.

At first sight, it is apparent that the number of bursts in a RS code-protected sequence has decreased significantly, while the length of these bursts has increased considerably relative to the convolutional-code-protected sequence and the no-correction sequence.

The behavior of the code is summarized in Table 4.6. The values presented on Table 4.6 are the mean result of a series of trials over all three test sequences.

The PSNR is linear with respect to the BER, as seen in Figure 4.13.

Like in the convolutional code case, λ_{error} increases as a function of the BER. This increase occurs because the concentration of errors increases with BER, and the number of errors bringing the RS

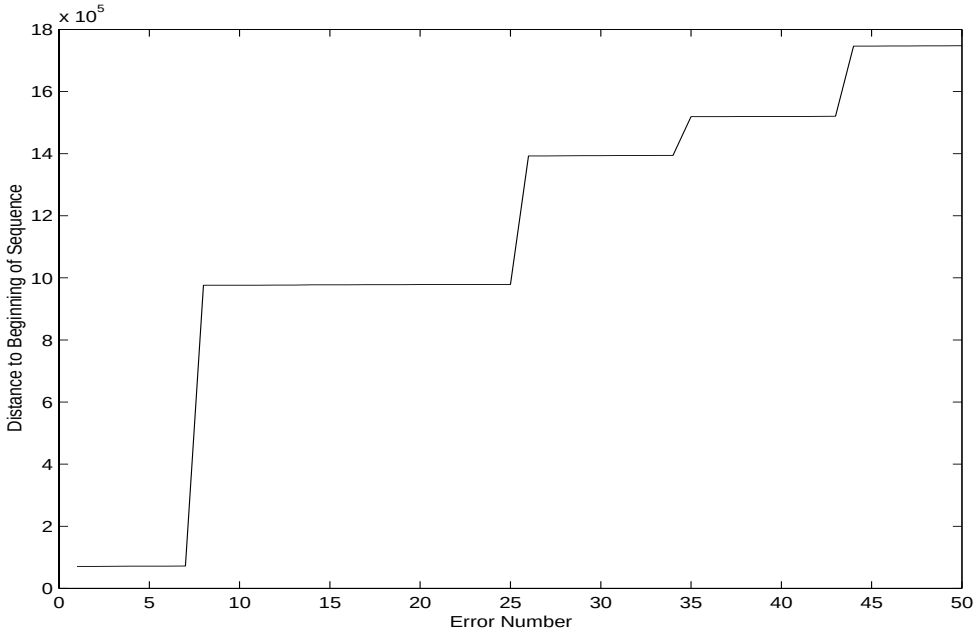

Figure 4.12: Bit Error Distribution, Reed-Solomon (255,239) Code (First Fifty Errors)

Table 4.6: Mean Behavior of Reed-Solomon Code for all Sequences

$\frac{E_b}{N_o}$	λ_{error}	$\lambda_{distance}$	BER	PSNR
6.5	9.4	4.48×10^4	2.03×10^{-4}	14.2
6.7	8.31	1.40×10^5	6.38×10^{-5}	19.1
6.9	8.82	5.63×10^5	1.62×10^{-5}	25.1
7.0	8.83	1.16×10^6	7.97×10^{-6}	29.6

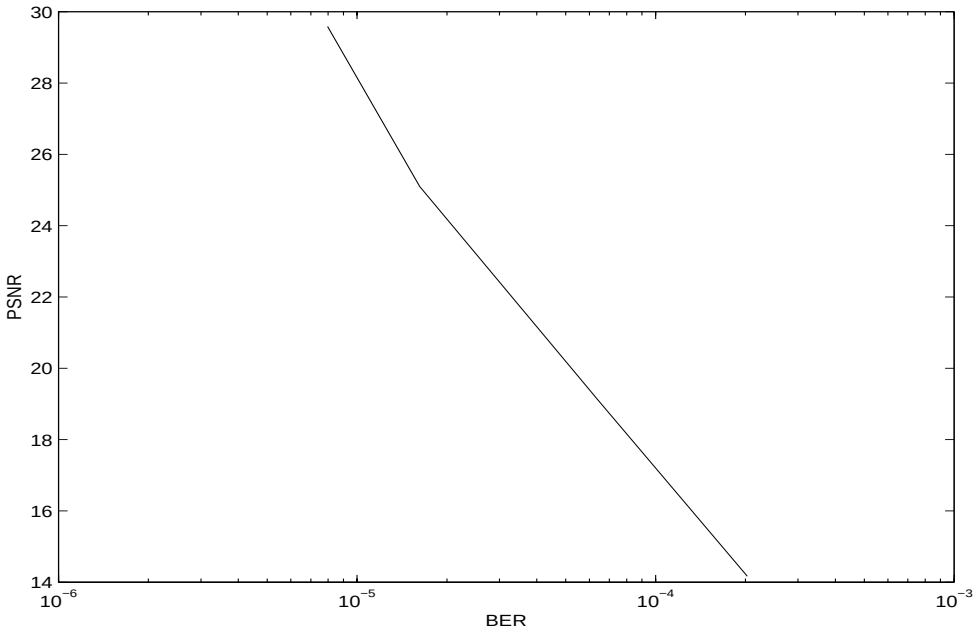

Figure 4.13: BER vs. PSNR for Sequence with Reed-Solomon-Code Protection

code past the failure points more often exceeds the threshold of nine errors.

An interesting point to observe, the value of λ_{error} is below nine for some cases. The error-correction capability of the code is eight, which means that there should *never* be any case in which the number of errors seen is less than or equal to eight. Eight or less errors in a packet can also occur when the redundant data added by the Reed-Solomon code has at least one error. The redundant data is discarded by the receiver once the RS decoder is passed. If there is an error in the redundant data (which would have brought the number of errors to a number higher than eight), then that error will not show up in the final error count, giving results that may mistakenly imply that there is a serious bug in the RS decoder.

4.5 Concatenated Code

This code combines the benefits of two powerful codes into one very powerful code. The RS and convolutional codes complement each other. While the convolutional code is designed to efficiently correct errors spread throughout the whole sequence, failing in the form of short bursts, the RS

code is designed to correct short bursts of errors.

Although the concatenated code as a whole was never validated, each piece making up the code has been validated. The RS code and the convolutional code were independently tested. The punctured convolutional code was also tested.

4.5.1 RS plus Convolutional Code

Figure 4.14 shows the bit-error distribution of a transmission in an AWGN channel when using the concatenated code. At first glance, it is apparent that the error burst length in this type of system is considerably longer than in the convolutional-coded, the RS-coded, or the no-coding case.

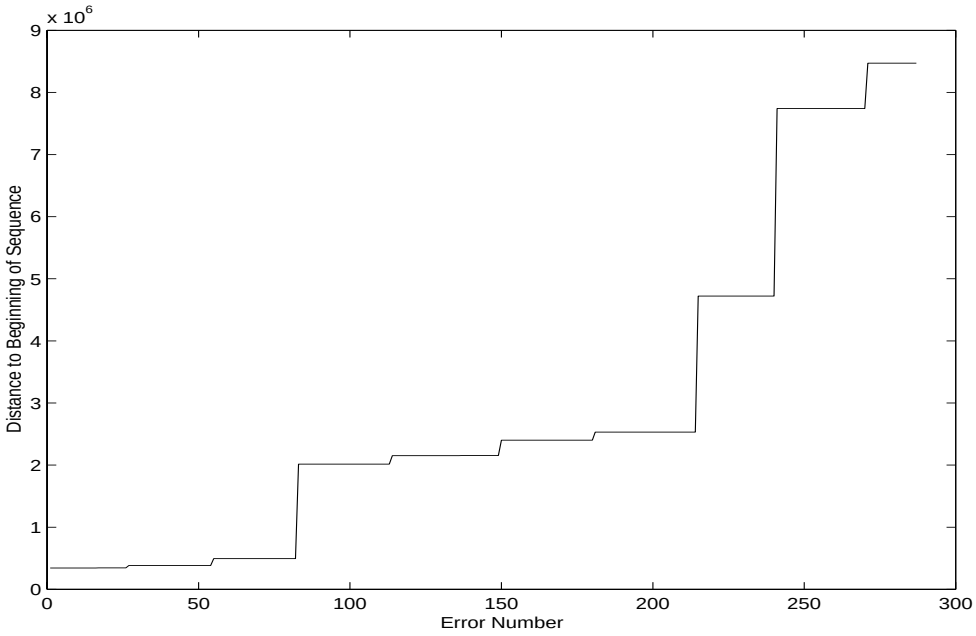

Figure 4.14: Bit Error Distribution, Concatenated $r = \frac{2}{3}$ Convolutional Code and RS Code

The concatenated code performance for several puncture rates is shown in Tables 4.7, 4.8, and 4.9.

4.5.2 Overall Code Performance

The BER-PSNR relationship for the concatenated code is given in Figure 4.15.

Table 4.7: Mean Behavior of Concatenated $r = \frac{1}{2}$ Convolutional and RS Code for all Sequences

$\frac{E_b}{N_o}$	λ_{error}	$\lambda_{distance}$	BER	PSNR
4.6	32.7	4.19×10^4	7.69×10^{-4}	15.0
4.8	31.9	1.19×10^5	2.62×10^{-4}	19.2
5.0	30.7	4.08×10^5	7.45×10^{-5}	23.9
5.1	30.7	7.28×10^5	3.80×10^{-5}	26.6
5.2	29.9	1.33×10^6	1.94×10^{-5}	29.4
5.3	28.9	2.68×10^6	8.79×10^{-6}	33.4

Table 4.8: Mean Behavior of Concatenated $r = \frac{2}{3}$ Convolutional and RS Code for all Sequences

$\frac{E_b}{N_o}$	λ_{error}	$\lambda_{distance}$	BER	PSNR
5.4	34.1	1.60×10^5	2.14×10^{-4}	20.6
5.6	32.4	4.46×10^5	7.24×10^{-5}	26.3
5.7	33.2	7.35×10^5	4.34×10^{-5}	30.0
5.9	32.4	2.24×10^6	1.21×10^{-5}	35.1

Table 4.9: Mean Behavior of Concatenated $r = \frac{3}{4}$ Convolutional and RS Code for all Sequences

$\frac{E_b}{N_o}$	λ_{error}	$\lambda_{distance}$	BER	PSNR
5.9	37.8	1.17×10^5	3.22×10^{-4}	19.7
6.1	37.1	4.98×10^5	1.37×10^{-4}	23.4
6.3	37.1	7.09×10^5	5.15×10^{-5}	28.4
6.5	36.8	2.42×10^6	1.70×10^{-5}	34.9

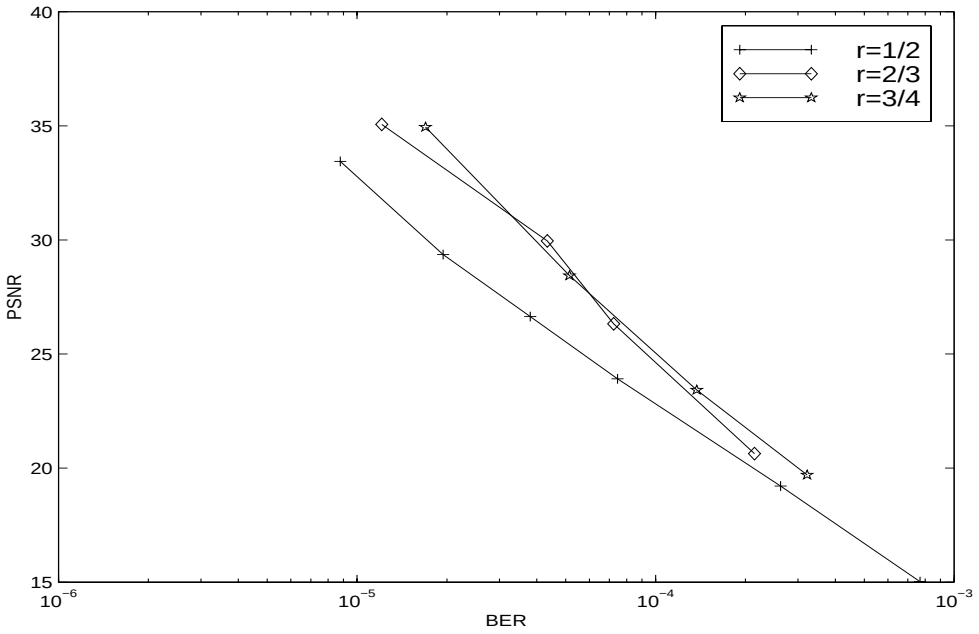

Figure 4.15: BER vs. PSNR for Sequence with Reed-Solomon-Code Protection

As seen in Figure 4.15, the relationship between the BER and the PSNR for all the sequences is roughly linear: the amount of decrease in PSNR is directly proportional to the increase in BER.

All configurations of the concatenated code presented the same effect on the λs: as BER increases, λ_{error} increases slightly, and $\lambda_{distance}$ increases considerably.

4.6 Co-Channel Interference

To further test the applicability of $\lambda_{distance}$ to video quality estimation and to understand the system's tolerance to interference, a set of trials with co-channel interference were simulated. Four systems were simulated: no correction, convolutional protection, RS protection, and concatenated protection (convolutional $r = \frac{3}{4}$).

Each system configuration was tested using different co-channel interference power levels, with $7dB \leq \frac{C}{I} \leq 20dB$ where $\frac{C}{I}$ is the carrier-to-interference ratio. The MPEG stream "alien" was used as the co-channel interference source. The co-channel interference source was sampled at a different rate than the main sequence, twenty-one samples per bit vs. ten samples per bit for the

main sequence. The intereference-generating signal was attenuated and added to the channel along with the Gaussian noise source.

The data collected in the non-Gaussian environment was collected on a single video sequence, 'susi.' Comparisons presented between Gaussian and non-Gaussian noise system performance are all based on data gathered when testing only the 'susi' sequence. The variance for these cases is high since only three trials for a single sequence were run instead of three trials for three sequences.

Co-channel interference testing centered on its effect on video quality with an emphasis on dissimilarities with a purely Gaussian noise source.

The performance of the system, as expected, degraded as a function of the amount of interference added to the channel. Figures 4.16 through 4.19 show the system BER for an unprotected, convolutional-, RS-, and concatenated- ($r = \frac{3}{4}$) protected sequence respectively.

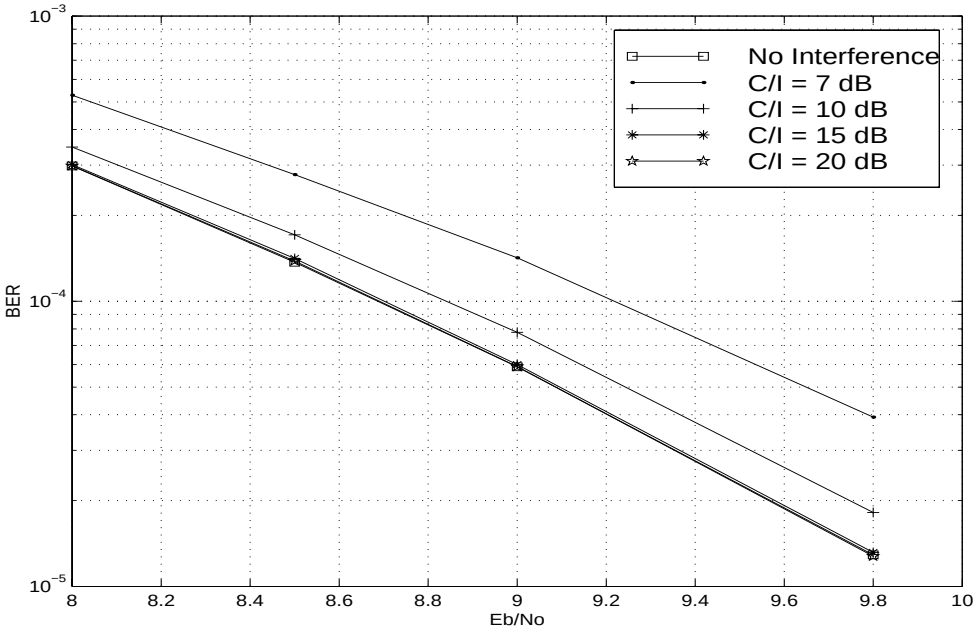

Figure 4.16: BER Performance of System with Co-Channel Interference and No Error Correction

As seen in the simulation results displayed in Figures 4.16 through 4.19, the BER is a function of both $\frac{E_b}{N_o}$ and $\frac{C}{I}$.

To see the effects of non-Gaussian noise on a video sequence, Figures 4.20 through 4.23 present

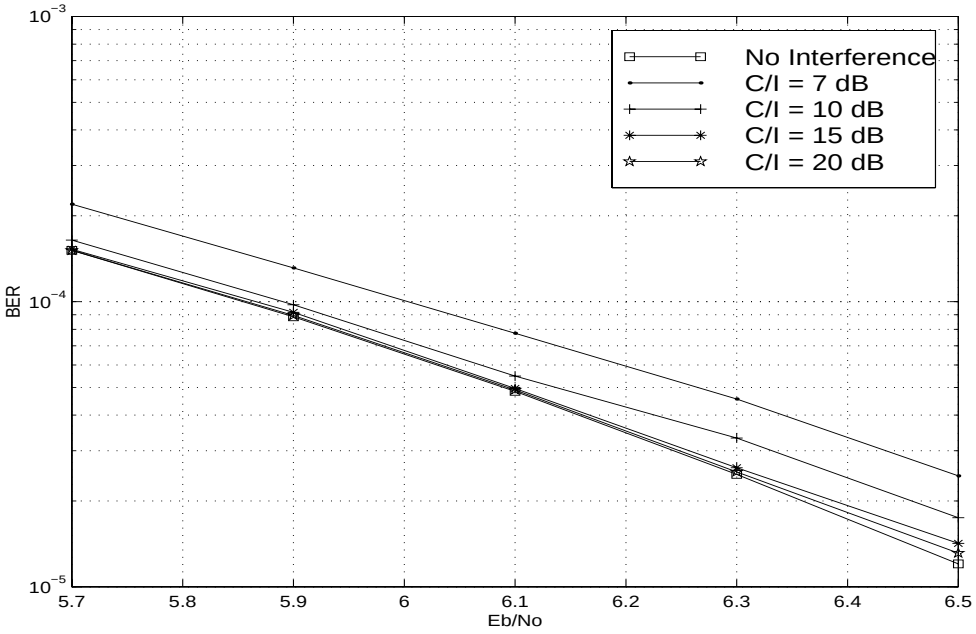

Figure 4.17: BER Performance of System with Co-Channel Interference and Convolutional Correction

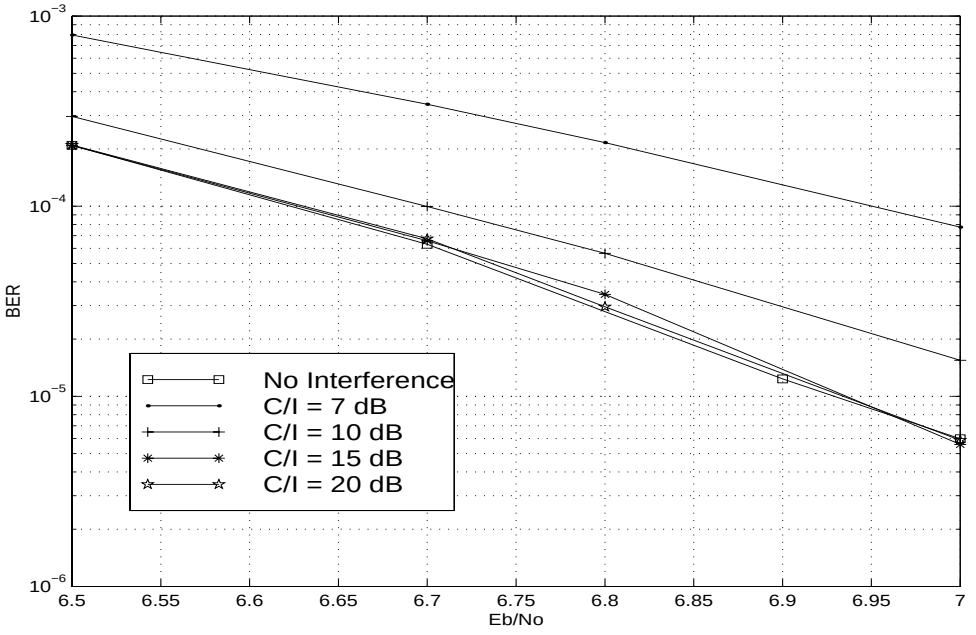

Figure 4.18: BER Performance of System with Co-Channel Interference and Reed-Solomon Correction

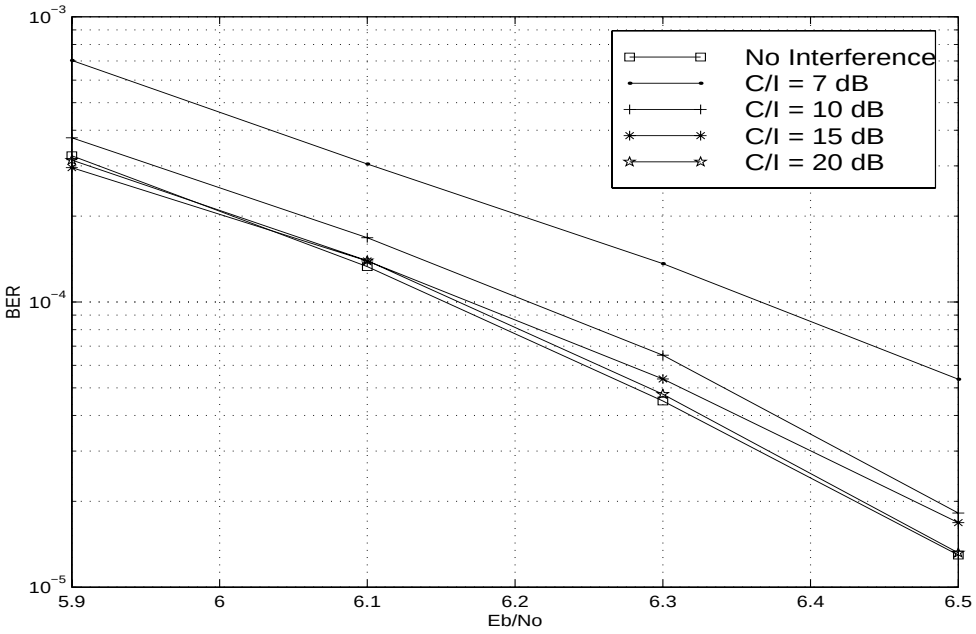

Figure 4.19: BER Performance of System with Co-Channel Interference and Concatenated Correction

BER vs. PSNR for the same cases as Figures 4.16 through 4.19. These graphs follow closely the system's performance in a purely Gaussian environment; video quality is a function of BER for each system regardless of the nature of the additive noise.

Figures 4.20 through 4.23 plot PSNR vs. BER for several error-correction formats. Note that even though the $\frac{E_b}{N_o}$-BER relationship changes as a function of $\frac{C}{I}$, the BER-PSNR relationship does not. Since $\lambda_{distance}$ can be derived from both BER and λ_{error} (Equation 4.3) and since λ_{error} does not change much as a function of $\frac{E_b}{N_o}$, then the $\lambda_{distance}$-PSNR relationship should match the BER-PSNR relationship. Figures 4.24 through 4.27 show that the linear $\lambda_{distance}$-PSNR relationship holds for the Gaussian and non-Gaussian noise scenarios.

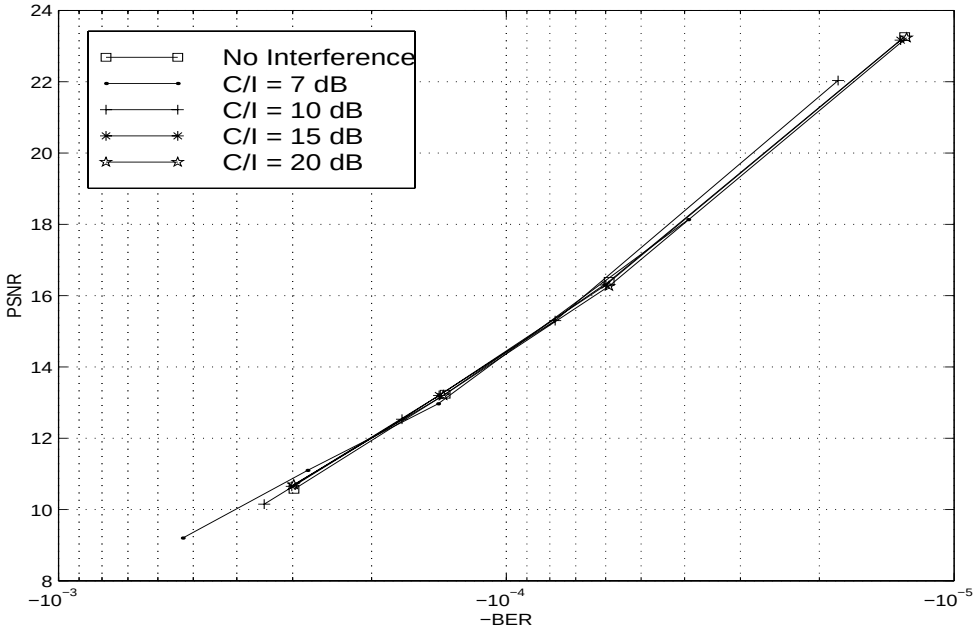

Figure 4.20: BER vs. PSNR of System with Co-Channel Interference and No Error Correction

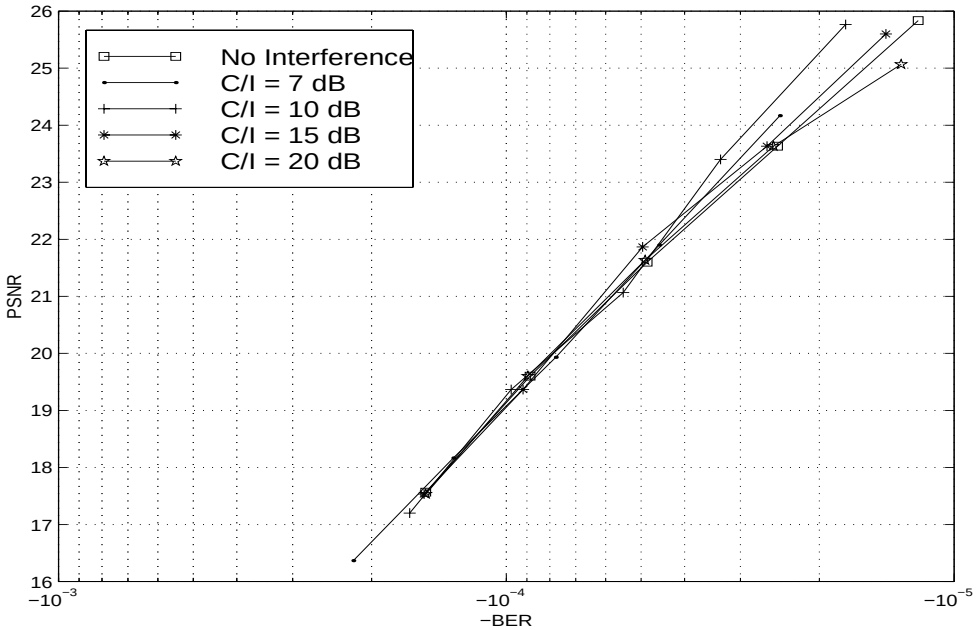

Figure 4.21: BER vs. PSNR of System with Co-Channel Interference and Convolutional Correction

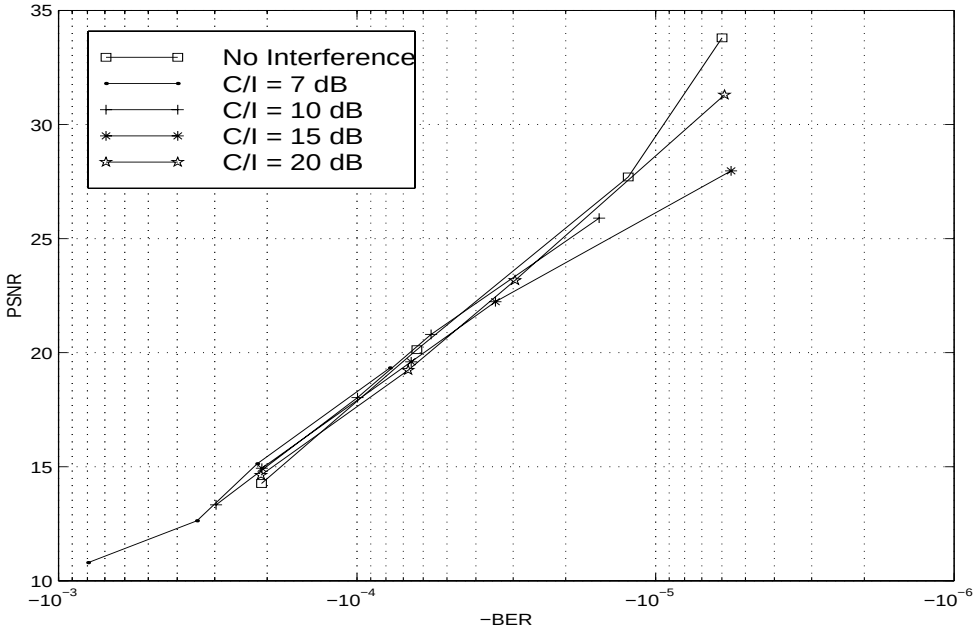

Figure 4.22: BER vs. PSNR of System with Co-Channel Interference and Reed-Solomon Correction

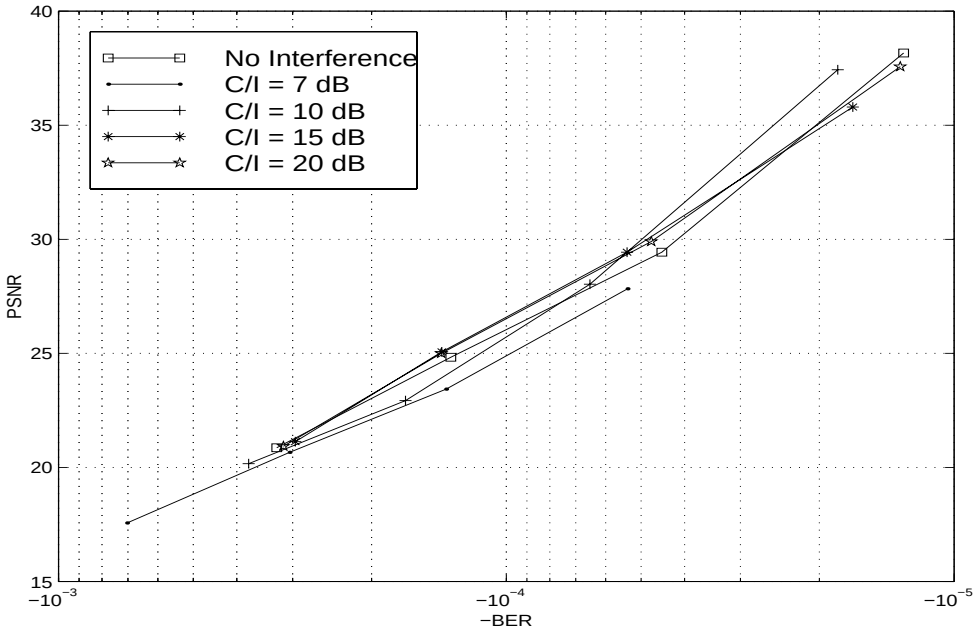

Figure 4.23: BER vs. PSNR of System with Co-Channel Interference and Concatenated Correction

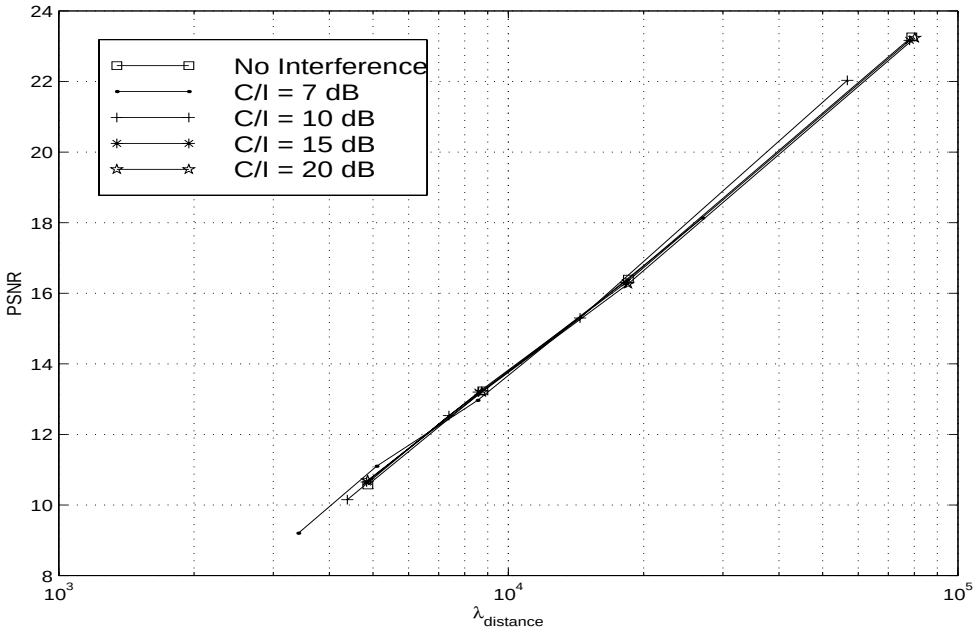

Figure 4.24: $\lambda_{distance}$ vs. PSNR of System with Co-Channel Interference and No Error Correction

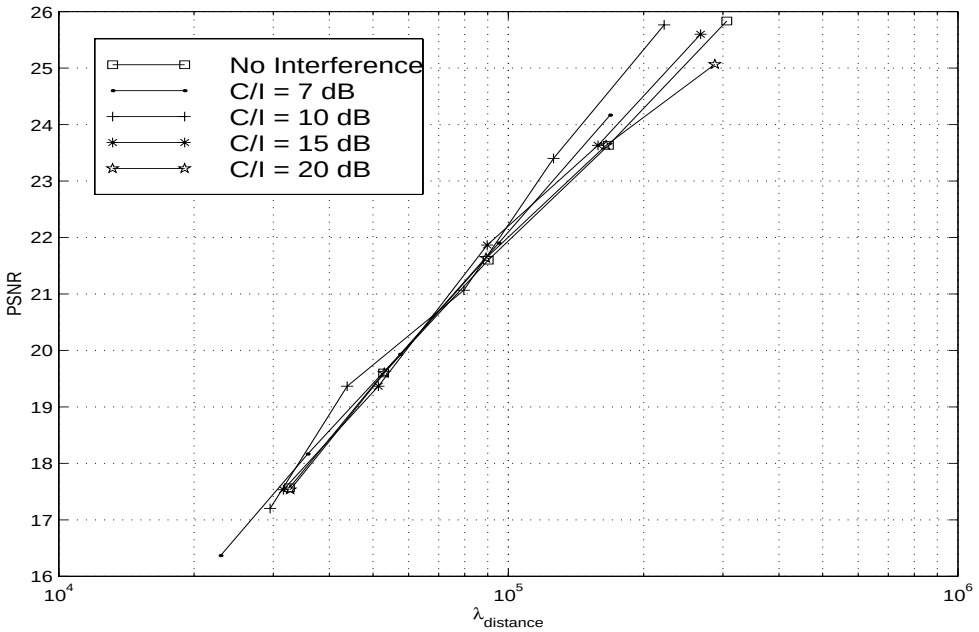

Figure 4.25: $\lambda_{distance}$ vs. PSNR of System with Co-Channel Interference and Convolutional Correction

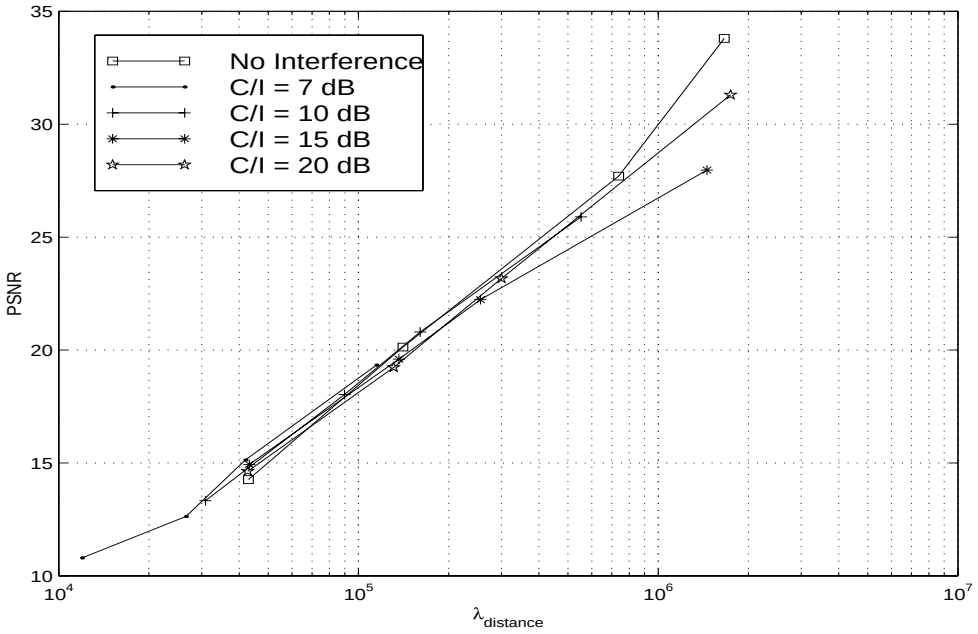

Figure 4.26: $\lambda_{distance}$ vs. PSNR of System with Co-Channel Interference and Reed-Solomon Correction

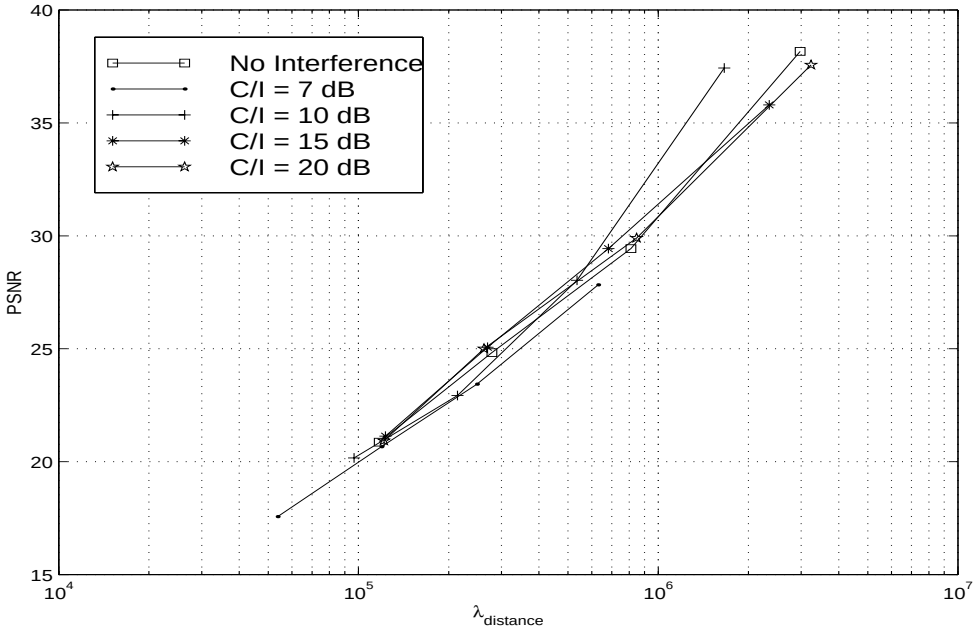

Figure 4.27: $\lambda_{distance}$ vs. PSNR of System with Co-Channel Interference and Concatenated Correction

Chapter 5

Results

MPEG video elementary stream is a dynamic data compression format. This format uses temporal and spatial redundancies to compress large amounts of visual information into a short data stream. This efficiency is offset by the high computational cost involved in the encoding/decoding process of an MPEG video elementary stream. The use of motion vectors and the process involved in the storage of a single DCT-block to reduce the amount of transmitted information requires large amounts of computer power at the encoder and decoder ends of a communication system. Since the method used for encoding (motion vector or compressed DCT) is a function of the data being encoded, determining the expected video quality solely from channel parameters is considered extremely difficult if not impossible by several experts in the field.

Though the evaluation of expected of video quality directly from channel characteristics has been an elusive goal, the following sections will present a statistical method that yields a single parameter derived directly from the channel that can be used to determine the expected video quality of a digital video sequence transmitted on a wireless channel.

5.1 Comparison of System Performance

In classical communications design, digital communications system quality performance is measured using BER. Parameters such as bandwidth, transmit power, antenna gain, and channel coding are

tweaked to not only fit the allocated resources (such as transmit power restricitions, mechanical antenna considerations, and available bandwdith) but also to minimize BER. BER is considered a sufficient metric to determine a digital communication system's performance regardless of data content.

While using BER blindly is the best way to determine system performance when the nature of the transmitted information is unknown, it is not sufficient to compare system performance when video is transmitted.

The data presented so far shows a linear relationship between BER and PSNR in a Gaussian and non-Gaussian channel. As the BER of a transmission increases, the PSNR decreases in proportion to the increase in the BER (as long as the video decoder algorithm does not collapse). This relationship is not sustained when the error-correction scheme is changed. Even though BER may be equal in systems with different error-correction schemes, their PSNR may be drastically different.

5.1.1 Assumptions

Three different MPEG video streams were simulated. However, to minimize variables, all streams were encoded using the same frame format (I-B-B-P-B-B-P-B-B-P-B-B-I). Note that the three sample sequences were selected as a function of their content: panning of a static background, panning of a scene with action, and a portrait with movement. Simulation results were consistent across all sequences.

Some assumptions that could affect results were used in the simulation. The size of the error-event window is defined as a single MPEG-2/TS packet. This decision is arbitrary and is not based on any video stream parameters. Also, the PSNR was assumed to be a sufficient metric of video quality. The PSNR has some limitations when comparing sequences of similar quality and evaluating the impact of some video artifacts.

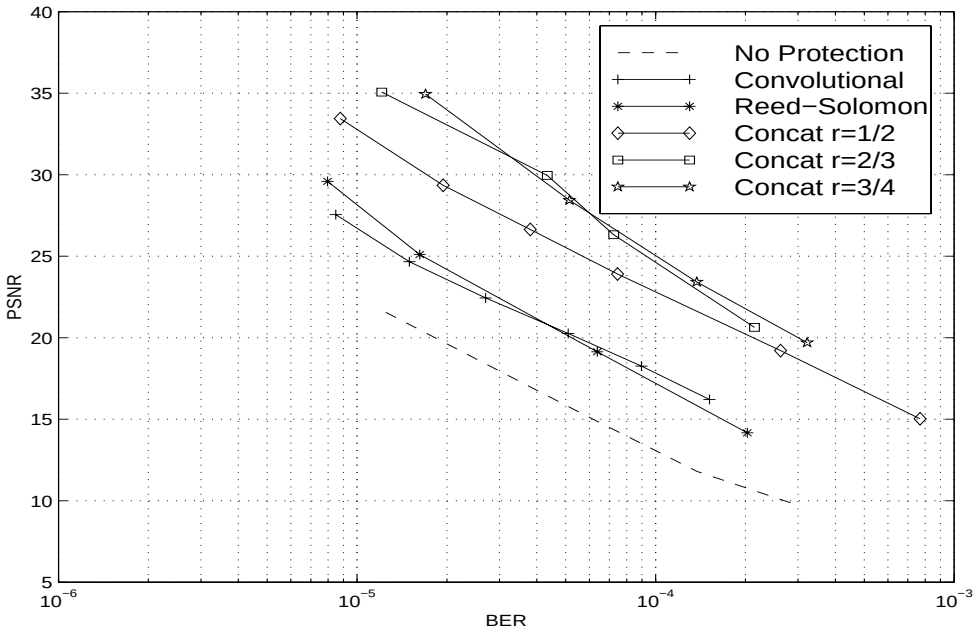

Figure 5.1: BER vs. PSNR for Gaussian Noise Cases

5.1.2 Data Analysis

Figure 5.1 shows a plot of BER vs. PSNR for trials run in an AWGN channel. This graph is designed to illustrate the effects of BER on video quality (PSNR) regardless of $\frac{E_b}{N_o}$; BER alone does not sufficiently describe the performance of a wireless video system. $\frac{E_b}{N_o}$ offers less guidance in predicting video quality than BER as seen in Figure 5.2. Clearly neither BER nor $\frac{E_b}{N_o}$ are sufficient channel metrics to accurately predict the transmission's video quality.

Though all curves in Figure 5.1 show a roughly linear relationship, some coding techniques offer better performance than others at similar BER levels. In other words, not all configurations yield the same value for video quality, quantified by PSNR, at the same BER. The difference is quite dramatic; there is a difference of more than 15 dB (PSNR) from the best to the worst performance at some BER lines. Even though the BER-PSNR curves may be offset by a large bias, they are all roughly parallel; their slopes are essentially the same.

The reason for this dramatic difference is the length of the error bursts. This observations supports the idea that the important factor when transmitting video is not just how many errors occur in a

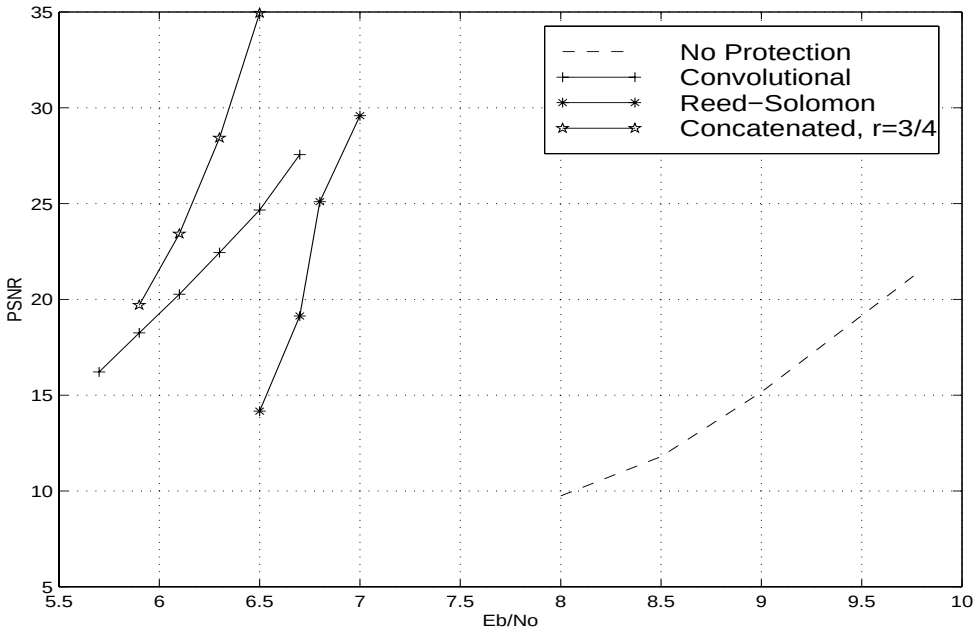

Figure 5.2: $\frac{E_b}{N_o}$ vs. PSNR for Gaussian Noise Cases

transmission but *their relative position and how frequently they occur.*

Table 5.1 presents several points in the performance curves extrapolated or interpolated (depending on the available data) for $BER = 1.2 \times 10^{-5}$. These values are taken directly from Figure 5.1. The curves seen in Figure 5.1 are roughly linear. For each curve that crosses the $BER = 1.2 \times 10^{-5}$ point, the PSNR value for that curve at $BER = 1.2 \times 10^{-5}$ can be interpolated. Some curves do not cross over $BER = 1.2 \times 10^{-5}$, but since there is a linear relationship, the expected video quality of that system at $BER = 1.2 \times 10^{-5}$ can be extrapolated. The extrapolated or interpolated video quality values at $BER = 1.2 \times 10^{-5}$ derived from the data plotted in Figure 5.1 are listed in Table 5.1.

Table 5.1 shows that there is a clear relationship between λ_{error} and the PSNR.

To understand why the relative positioning of the errors matters, it is critical to have a basic understanding of how digital video (in this case, MPEG-2) is encoded.

Each frame of video is broken up into blocks of eight pixels by eight pixels. Each of these blocks is converted to the frequency domain using a Discrete Cosine Transform (DCT). The DCT coefficients

Table 5.1: Selected System Performance Points

Method	λ_{error}	Extrap/Interp PSNR (at 1.2×10^{-5})
None	1.02	21.8
Conv $r = \frac{1}{2}$	3.98	25.8
RS	8.82	27.0
Concat $r = \frac{1}{2}$	29.9	31.8
Concat $r = \frac{2}{3}$	32.4	35.1
Concat $r = \frac{3}{4}$	36.8	36.9

are quantized (zeroing out the higher order coefficients), and the resulting DCT is entropy-coded (Huffman code). If a single bit error were to occur somewhere in this DCT as it is transmitted, the corrupted data is frequency information; the inverse-DCT is applied and the time-domain data is recovered resulting in a corrupted block. In other words, the DCT is very fragile and susceptible to a single bit error. This type of error appears in the decoded video sequence as an off-color block on the screen, a very large form of salt-and-pepper noise, an example of which is seen in Figure 5.3.

DCT-coded blocks are arranged in slices, which compounds decoding problems. Blocks are assembled into Macroblocks (a collection of different blocks representing the color vectors for the same general area). These macroblocks are built and sent as a linear sequence of blocks. Figure 2.14 presents the layout of blocks within a macroblock. The slice is composed of a sequence of macroblocks. If a single bit error occurs on a block, the Huffman coded sequence may result in a bit insertion or bit deletion. The insertion or deletion of bits in the uncompressed sequence will result in a loss of synchronization in the block decoding, and the rest of the slice will be lost. The next slice is not corrupted because the decoder finds the next start code corresponding to the beginning of a slice. In short, a single bit error can corrupt a large part of several rows of a frame. If this frame is an I or P frame, this error will propagate until the corrupted frame is no longer used as a reference. An example of this type of error is seen in Figure 5.4. Note that this error will vary as a function of the decoder implementation. In this example, the memory buffers are not reset before the construction of each new frame, with the result that old data will be seen in areas where the decoder failed. On the other hand, if the decoder cleared its memory buffer before each frame is

Figure 5.3: Corruption of Individual Blocks

built, a failed slice would appear black where the decoder failed.

Another entity in the video transmission that is just as vulnerable to bit-errors as the DCT is the motion vector, a single number that determines the relative position of blocks. If a single bit on a motion vector is damaged, the calculated position of the block is damaged beyond repair. An example of corrupted motion vectors is seen in Figure 5.5. This type of error appears on a decoded sequence as the appearance of a feature where it should not be seen. The example seen in Figure 5.5 shows flowers a third of the way up the trunk of the tree on the right side of the frame.

In all of these cases, the effect of a single bit-error is catastrophic. Therefore, by extension, if one bit error is catastrophic, several bit-errors are also catastrophic. If the visual effect of a single bit-error has the same as the effect of several bit-errors, clustered errors will look as if the total number of bit-errors is much lower than it really is; this is where λ_{error}, a measure of clustering, becomes important. Since λ_{error} can be used as a measure of comparison of clustering between different systems and since BER is a good measure for comparing the performance of the same system under different levels of noise, a combination of λ_{error} and BER gives a good indication of

Figure 5.4: Corruption of Whole Slices

Figure 5.5: Corrupted Motion Vector

overall system performance.

Table 5.1 supports this postulation about the inherent connection between PSNR and mean event size (λ_{error}). Table 5.1 also presents evidence that supports using λ_{error} as a metric to determine the relative performance of different systems when their BER is equivalent. Note that while BER remains constant as PSNR decreases, λ_{error} accurately reflects the PSNR trend. Since BER is a factor in the received video quality, a metric based on both λ_{error} and BER should provide an absolute measure of video quality.

As Equation 4.3 shows, $\lambda_{distance}$ is a function of both BER and λ_{error}. $\lambda_{distance}$ is the mean distance between error-events. Figure 5.6 presents a plot of $\lambda_{distance}$ vs. PSNR for simulations in an AWGN channel.

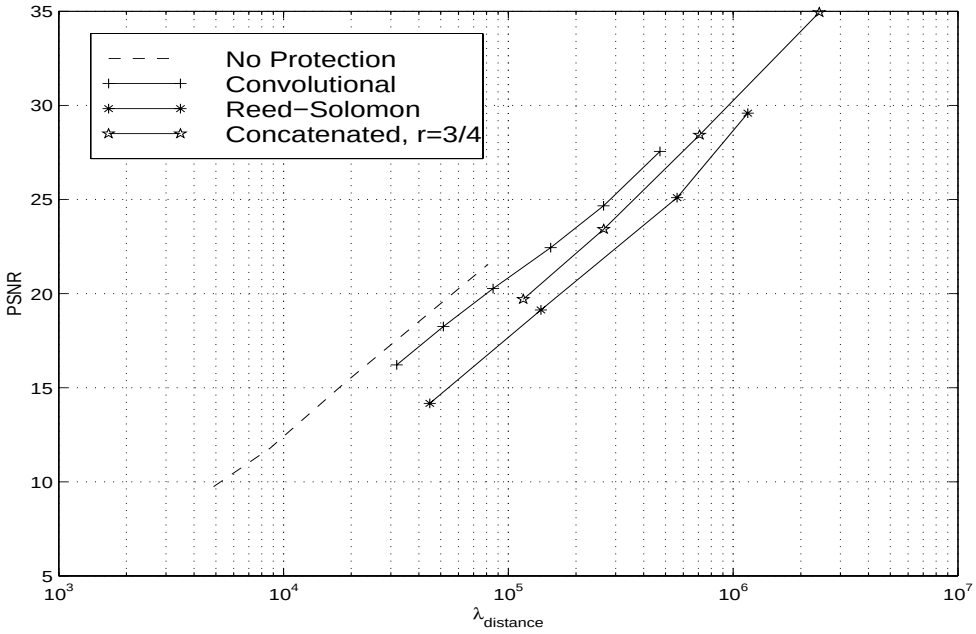

Figure 5.6: $\lambda_{distance}$ vs. PSNR for Cases in AWGN Channel

Figure 5.7 presents the performance of different systems evaluated for an AWGN channel; it shows a distinct linear relationship between BER and PSNR for each system. This relationship provides a basis for comparing video quality for equivalent BER values across systems. Note that, in general, systems with the same BER do not provide equivalent video performance. There is no absolute relationship between PSNR and BER; the BER only offers a relational level of quality for a par-

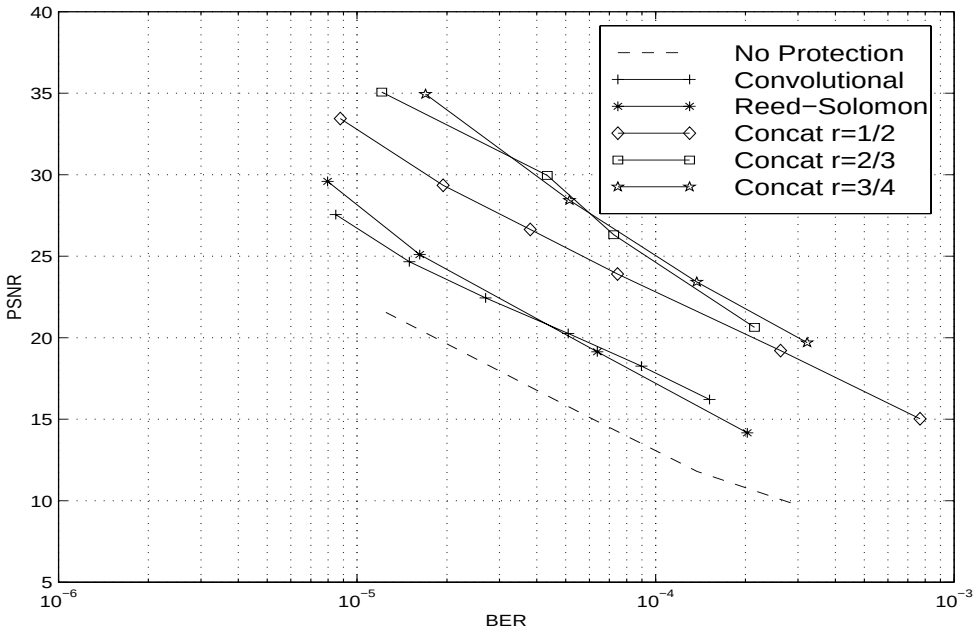

Figure 5.7: BER vs. PSNR for All Cases in AWGN Channel

ticular system. No BER value can be *always* linked to a single PSNR value. Figure 5.7 shows that a single BER value is associated with a PSNR range of over 15 dB depending on the system. BER offers only a relative measure within a single system setting. The only video quality-prediction information available based solely on BER is a relative measure of PSNR; a lower BER means a higher PSNR. No information regarding gains over other system configurations is provided solely from BER.

Figure 5.6, on the other hand, offers an absolute video quality measure. Figure 5.6 shows a series of closely set lines. Each line represents a plot of $\lambda_{distance}$ vs. PSNR. Important to note, in Figure 5.6, just like in Figure 5.7, all curves are fairly linear. However, unlike Figure 5.7, Figure 5.6 shows that curves all line-up very closely (± 2 dB). The spread around the mean value begins to diverge at around 10 dB PSNR. This divergence occurs because of inaccuracies in the calculation of λ_{error} at low BERs, seen as a slight increase in the value of λ_{error}, and because at the higher BERs the video decoder is very close to total decoding failure.

Though two systems have the same $\lambda_{distance}$ and similar PSNR values, their respective BERs may vary by as much as an order of magnitude. Unlike BER, $\lambda_{distance}$ is capable of predicting video

quality in a wireless system based solely on physical layer parameters. Figure 5.8 shows the BER-$\lambda_{distance}$ relationship. There is a linear relationship between the two parameters, with the curves that make up the chart being separated by a large bias. This bias is such that curves with equivalent $\lambda_{distance}$ values can be offset by a difference of an order of magnitude in BER.

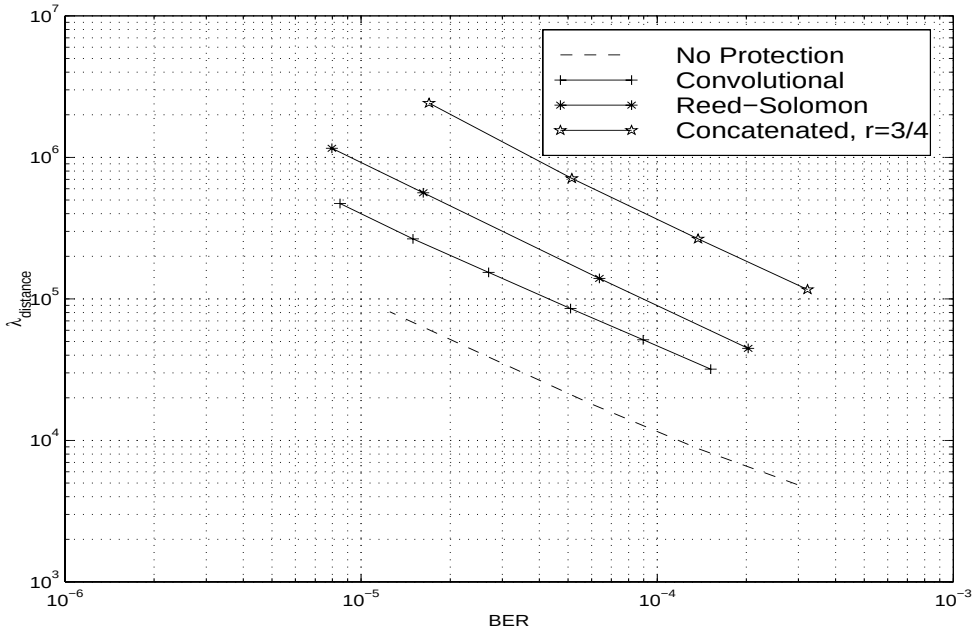

Figure 5.8: BER vs. $\lambda_{distance}$ for Cases in AWGN Channel

The data presented so far supports the use of $\lambda_{distance}$ as a physical layer metric that can yield an absolute PSNR prediction for a video transmission in an AWGN channel. Chapter 4 presented data collected in a non-Gaussian channel. Note that since the non-Gaussian channel was tested for the 'suzi' sequence only, the results have to be compared with only the 'suzi' from the Gaussian trials.

Figure 5.9 shows the BER-PSNR relationship for the uncoded, convolutional-, RS-, and concatenated-coded systems with different levels of co-channel interference. Note that Figure 5.9 is similar to Figure 5.7. The curves describing systems with co-channel interference stay close to the curve describing the system with AWGN noise only.

Figure 5.10 presents the $\lambda_{distance}$-PSNR relationship for the unprotected, convolutional-, RS-, and concatenated-coded systems. Similar to Figure 5.9, the curves on Figure 5.10 representing systems

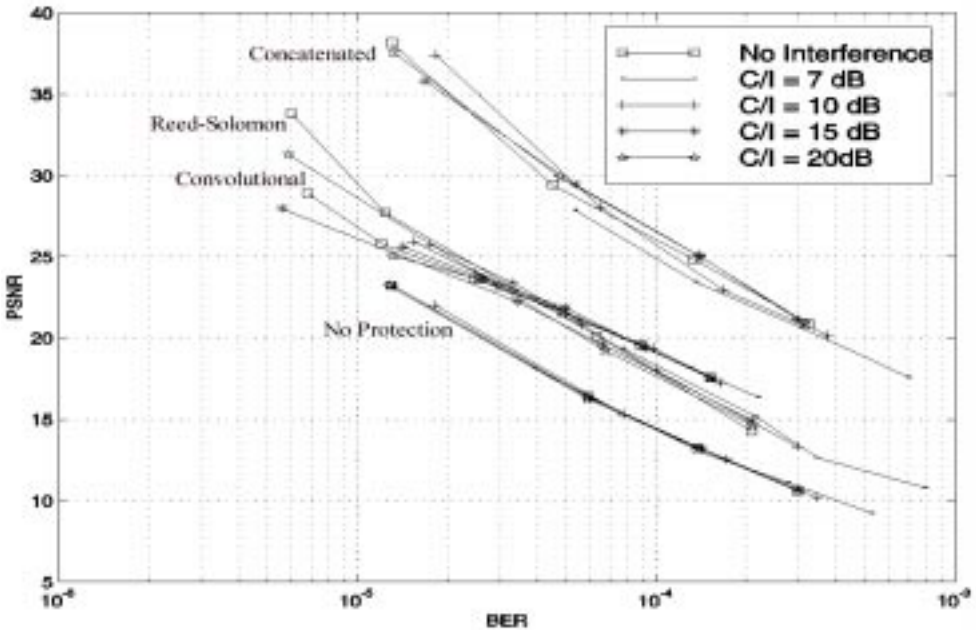

Figure 5.9: BER vs. PSNR for Cases with Co-Channel Interference

with co-channel interference stay close to the curve representing the same system with AWGN noise only.

The curves in Figure 5.10 closely follow (within $\pm 2dB$) a center line. Figure 5.10 is very similar to Figure 5.6, leading to the conclusion that $\lambda_{distance}$ is a good parameter for the prediction of expected video quality in a wireless transmission.

Furthermore, the relationship between BER and $\lambda_{distance}$ seen in Figure 5.8 still holds for the co-channel interference case as seen in Figure 5.11. This relationship is logical since $\lambda_{distance}$ is essentially the BER curve with a fixed correction of λ_{error}.

It is difficult to provide a human-based metric that corresponds to video quality, i.e., evaluating the accuracy of PSNR is difficult. There is a certain amount of correlation between subjective video quality and PSNR. At close levels (a couple of decibels), the PSNR metric is difficult to evaluate; sequences of similar quality may receive PSNR values differing by a few decibels. However, PSNR proved accurate when comparing sequences with large quality differences (over 10 dB). When the video quality of two sequences is noticeably different, PSNR will give the better-quality sequence the higher value. This means that the BER-PSNR graphs, which show a large PSNR

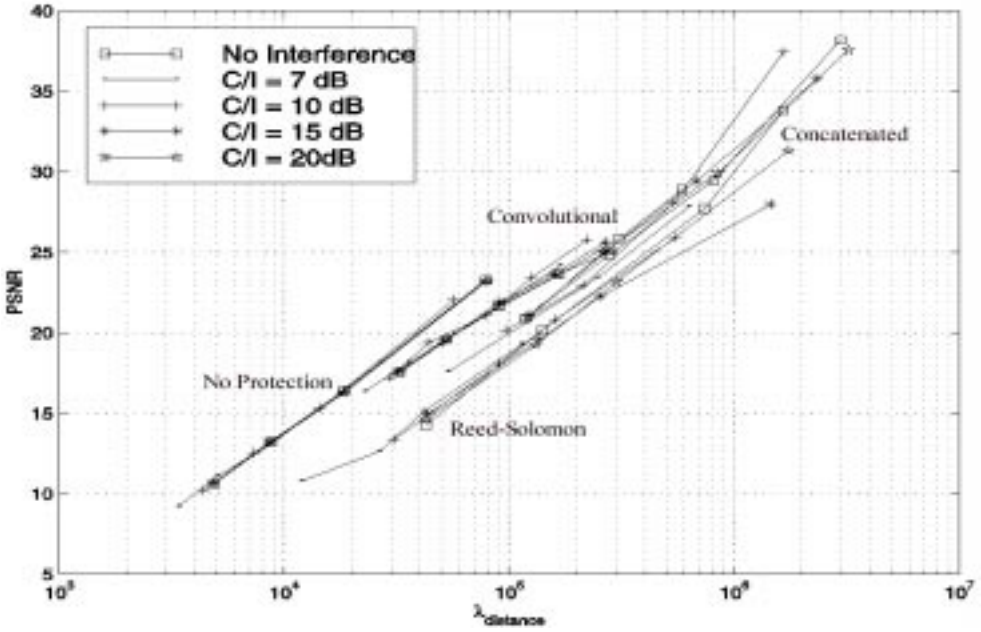

Figure 5.10: $\lambda_{distance}$ vs. PSNR for Cases with Co-Channel Interference

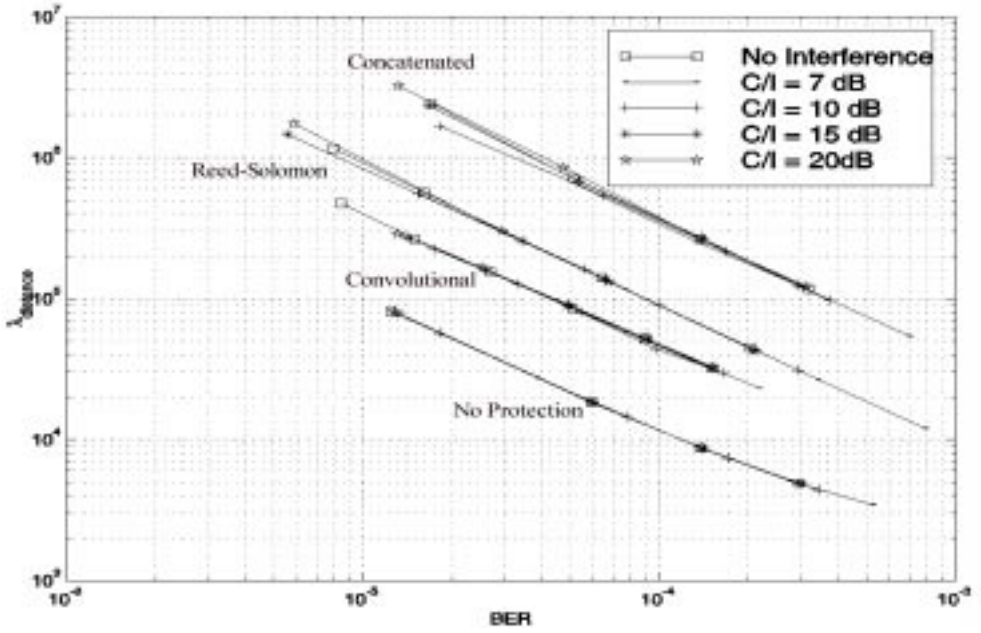

Figure 5.11: BER vs. $\lambda_{distance}$ for Cases with Co-Channel Interference

spread between different systems at equivalent BER values, really represent large quality differences between sequences with the same BER. The $\lambda_{distance}$-PSNR curves, on the other hand, have a spread of only a couple of decibels; this means that difference in quality between different video sequences with the same $\lambda_{distance}$ was not noticeable in the observed trials.

Note that there are several problems associated with this approach to system analysis. The metrics used to determine system performance are not the most reliable sources of information. The PSNR cannot evaluate temporal effects, and it is best employed when comparing sequences with large differences in quality. Since the metric used for deriving video quality is not a very accurate measure, any metrics derived from the use of PSNR, namely λ_{error} and $\lambda_{distance}$, leave a certain measure of uncertainty in their results.

Another problem inherent to the λ_{error} method of predicting video quality, the algorithm used to determine event boundaries is not based on the video encoding scheme. Since a single bit-error has the ability to corrupt features as small as a motion vector and as large as a whole slice, it is essentially impossible to determine the best size of an error burst. The only feature of the system that is constant is the size of an MPEG-2/TS packet; this is the size that was selected as a burst event. A single packet may include several objects, thus what may, by definition, be a single error burst event may really affect several objects in a video sequence. Since objects in a video sequence can also be very large (such as a whole slice), multiple packets may make up a single object; what the algorithm considers multiple bursts may appear in the video sequence as a single feature. In other words, the ideal error-event detector algorithm should be responsive to the video transmission, error-events should be judged on the video data it is affecting. However, because of the complexity of this approach, a fixed error-event window was determined based solely on the communications system packet size. It is hoped that by selecting a fixed error-event window in which to find an error-event, the short-term differences in video feature length can be averaged out and the metric will converge into a meaningful value.

Approaches based on corrupted symbols [21] have been suggested. The analysis of system performance based on corrupted symbols is based on the corruption of a bit on a single symbol, like a byte, resulting in a corrupted byte. Therefore, the effect of a single bit-error is the same as eight bit-errors. System performance bounds can then be analytically calculated based on the maximum

and minimum effects of an error burst on a system. This problem increases considerably in complexity once the symbols are not fixed-length. Video coding is composed of variable-length objects, which can be interpreted as variable-length symbols.

Even though there are problems inherent to the calculation of the λ parameters fit, the presented data supports a strong relationship between $\lambda_{distance}$ and PSNR.

5.2 $\lambda_{distance}$ Applications

Several applications of this new figure of merit are possible, an example of which is adaptive video applications. In a wireless environment, the channel is a dynamic, time-varying medium. The Quality-of-Service (QoS) that can be offered by a wireless system can change quickly because of environmental conditions. Adaptive transmission parameters such as data rate, coding, and spreading gain, can be varied to support a desired QoS.

$\lambda_{distance}$ is a parameter that offers the possibility of real-time video quality assessment at the physical layer without the need of decoding the transmitted traffic. The value of $\lambda_{distance}$ can be calculated with the use of a training sequence. However, there is a problem with the training sequence approach. Acceptable video quality is application-dependent. Quality that may be acceptable for some surveillance applications or video-phone may be completely unacceptable for entertainment purposes. A PSNR of 25 dB offers video quality that may be acceptable for some applications, such as surveillance, which requires $\lambda_{distance} \approx 4 \times 10^5$. To determine if $\lambda_{distance} \approx 4 \times 10^5$, the training sequence would have to observe around ten error events totalling 4×10^6 bits for a single observation. Multiple observations would raise the training sequence to tens of megabits, an unacceptably large length.

There is another way to calculate $\lambda_{distance}$ rather than using a training sequence. Refined methods exist for quickly determining the SNR or $\frac{E_b}{N_o}$ of a system [22]. Since the system configuration is known, BER can be estimated quickly by using a chart like Figure 5.12.

Since the system configuration is known, not only is a $\frac{E_b}{N_o}$-BER chart available, but a BER-$\lambda_{distance}$ chart like Figure 5.13 is also available.

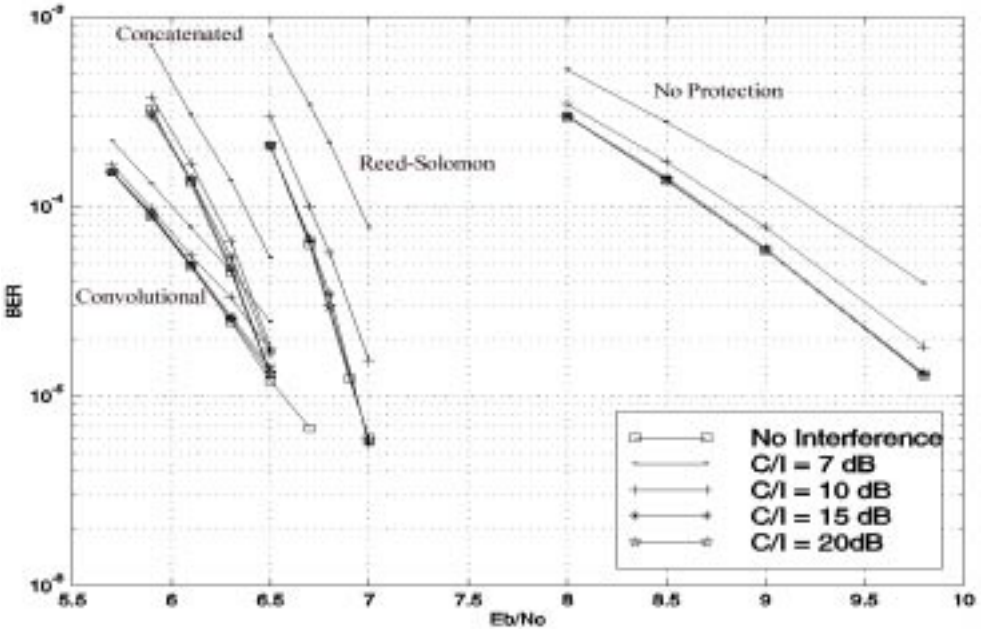

Figure 5.12: $\frac{E_b}{N_o}$ vs. BER for Cases with Co-Channel Interference

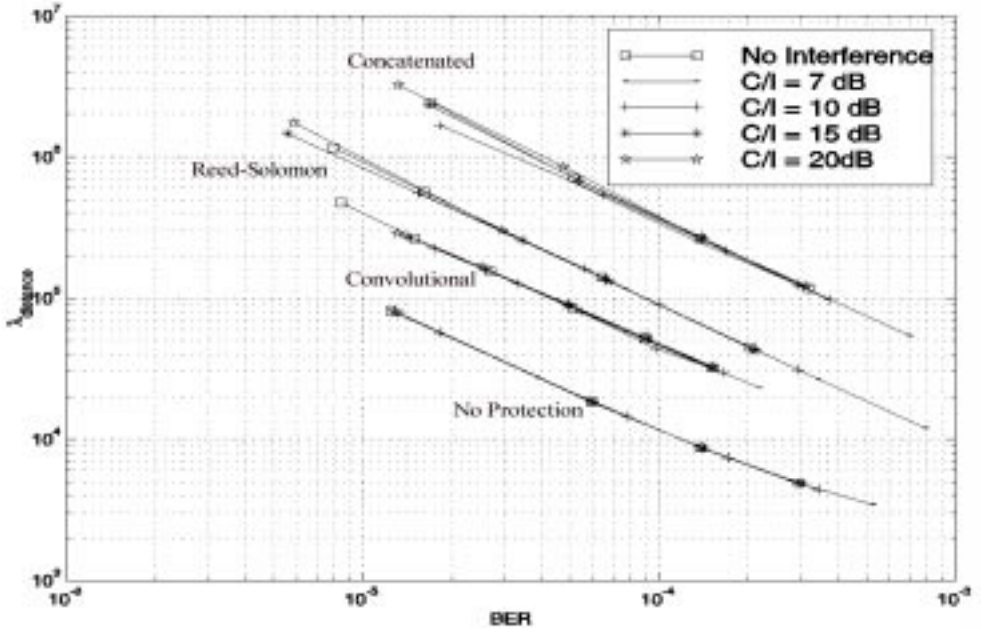

Figure 5.13: BER vs. $\lambda_{distance}$ for Cases with Co-Channel Interference

Figure 5.14 presents a $\frac{E_b}{N_o}$-$\lambda_{distance}$ graph. If the $\frac{E_b}{N_o}$ of a system is calculated, $\lambda_{distance}$ can be derived directly from Figure 5.14.

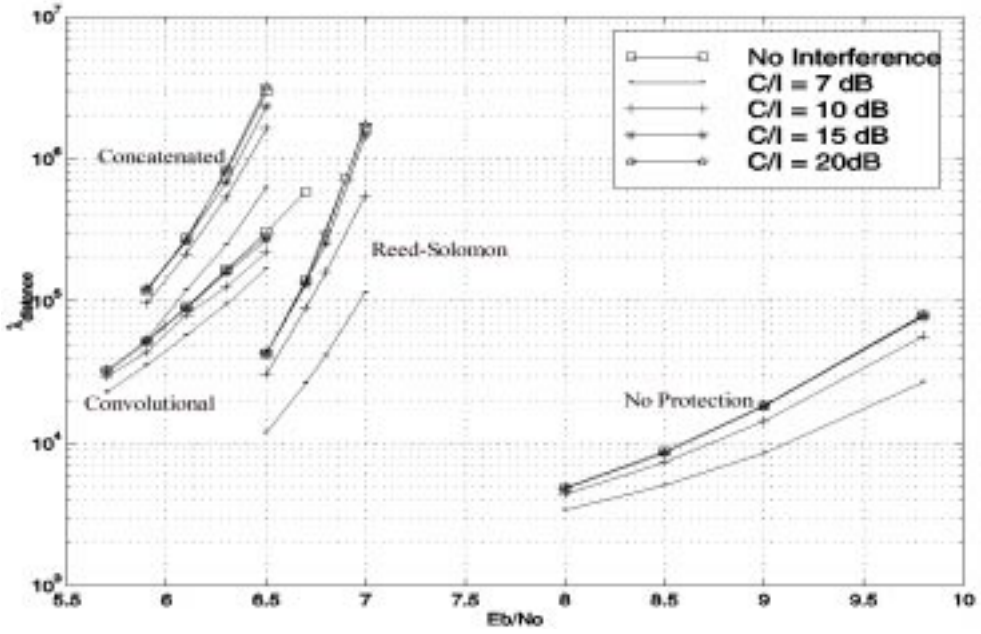

Figure 5.14: $\frac{E_b}{N_o}$ vs. $\lambda_{distance}$ for Cases with Co-Channel Interference

Using either Figure 5.12 and Figure 5.13 or their combination seen in Figure 5.14, all of which were calculated in simulation and stored in the system's memory, the $\lambda_{distance}$ value needed to achieve optimal video quality for the application can be quickly calculated.

5.3 Conclusion

The BER is a basic metric for digital communication system design. This metric is a valuable method in determining the performance of a system when the data is of unknown format. However, it is not necessarily the best metric when comparing different systems supporting a known application such as the transmission of digital video.

Typical users of broadcast video applications, such as television, expect a high level of video quality. For such users, the digital video resulting from the transmission of a sequence in a high-BER environment will yield an unacceptable level of distortion. This distortion is in the form of large

artifacts appearing on the video display. In this thesis, the transmission of digital video in a high-BER environment was studied. As a measure of video quality, the relative number of artifacts appearing on an image was measured using the PSNR. Although the definition of video quality used in this thesis does not apply to broadcast television applications, it does serve as a measure of the usability of a video sequence. In some applications, such as surveillance, the occasional appearance of artifacts may be acceptable to the user. In such applications, the relative number of artifacts is important, and it is with those applications in mind that the term "video quality" is used in this thesis.

We showed that video transmissions are highly susceptible to channel noise. The effect of the channel noise is such that a single bit error will become apparent to the user. The effect of a single bit-error is a catastrophic event at a localized portion of the video. Further, the addition of more bit-errors in the same local area will have no additive effect on the apparent video quality. Hence, if a particular block is corrupted by a single bit-error, several errors added to the already corrupted block of data will have no visible effect.

If bit-errors are an unavoidable event, then it is important to cluster the errors in a tight unit to minimize the visual impact of these errors on the video sequence. A cluster of errors, when viewed as a single event, can be described as a Poisson counting process. One can completely describe a Poisson counting process by the value λ, the mean-arrival rate.

The metric λ_{error}, the mean burst length, was proposed as a guideline supplementing the BER in determining the expected performance of a wireless communication system transporting digital video.

Through the use of simulation, this thesis demonstrated the strong relationship between λ_{error} and video quality for signals with similar BER. This relationship was extended through the use of the error-event-mean-arrival-rate, $\lambda_{distance}$, a value that one can derive from both λ_{error} and BER. The primary value of $\lambda_{distance}$ is its ability to predict video quality based entirely from physical layer parameters. $\lambda_{distance}$ is a parameter that proved capable of predicting video quality based solely on channel performance regardless of the error correction method used. An exhaustive literature search revealed no references for the use of $\lambda_{distance}$ or any other metric used to calculate an absolute prediction of video quality based solely on channel parameters.

An application for real-time adaptive video systems to monitor QoS was suggested using $\lambda_{distance}$ calculated from apriori-knowledge of system layout and expected performance based on $\frac{E_b}{N_o}$ values. Although more study is needed to determine the breadth of applicability of this metric to different MPEG coding settings, simulation has proven the existence of a strong correlation between $\lambda_{distance}$ and PSNR. This metric presents a fundamental addition to the field that was not available prior to the research presented in this thesis.

Chapter 6

Conclusion and Suggested Further Work

The relationship between video quality, video coding, channel coding, and channel characteristics has been investigated in this thesis. It is found that the mean distance between bit error clusters, $\lambda_{distance}$, which can be derived directly from the physical layer, makes possible the prediction of video quality. In this thesis, video quality was measured as the relative number of artifacts appearing in a video sequence. Although the appearance of artifacts in a sequence is not acceptable in applications such as broadcast television, it may be acceptable in applications such as surveillance. The parameter derived in this thesis can be used to predict the mean Peak-Signal-to-Noise-Ratio (PSNR) value of a video sequence without needing to decode a video sequence. The use of this parameter is important because it allows for quick and efficient algorithms needed in applications maintaining Quality of Service (QoS).

Variable QoS systems are a necessary development for the deployment of wireless personal video communication devices. Given the dynamic nature of the wireless channel, consistent delivery of acceptable video quality is a problem. Algorithms are needed that can quickly react to changing channel conditions and optimize the architecture for the given conditions. The research presented in this thesis reveals a general statistical method, centered on the metric $\lambda_{distance}$, which can quickly estimate video quality without requiring the system to decode and evaluate transmitted video data.

The data in this research was collected using a software tool designed to study the effects of a wireless channel on a video transmission. This tool allows the simulation of a flexible communication system architecture in conjuction with several standard packet data formats. The simulation tool can also simulate several fading environments and interference issues, such as co-channel and adjacent channel interferors.

Several aspects of digital video wireless transmissions were not studied, namely the effects of a fading envelope. A few papers presenting the results of fading channels on video transmissions have been published. A fading channel is common in mobile communications. However, the effects of a fading channel coupled with different error-correction techniques and framing format on a video transmission is a subject for further study.

The bulk of errors seen in the simulated video sequences stem from corrupted slices and the inability of the MPEG decoder to re-synchronize the decoder and recover uncorrupted data blocks within a slice with corrupted blocks. System performance would improve considerably if a method were developed that allowed the recovery of data unavailable due to synchronization problems.

No bounds were calculated for the performance of the system. Previous work on the effects of clustered bit-errors on received fixed-length symbols [21] has led to the derivation of tight performance bounds. The work presented in this thesis deals with clustering of errors on received variable-length symbols (encoded video objects). Future work should examine system performance bounds using the approaches presented in [21].

Furthermore, the video sequences used shared the same frame format (I-B-B-P-B-B-P-B-B-P-B-B-I). Given that MPEG is a rich format that allows a variety of coding configurations, further work is suggested for studying the applicability of $\lambda_{distance}$ when using a variety of MPEG coding schemes to better determine the limits and capabilities of this metric.

MPEG-2 is not the only standard available for video coding. MPEG-4, the H.26x family of video coding formats from the ITU, and wavelet-based video coding are also formats that are different from MPEG-2, and their needs may not be served by the $\lambda_{distance}$ metric. Further work concerning the interaction between channel coding and alternate video formats is also suggested.

Appendix A

Packet Format

The MPEG format is a powerful video coding format that is flexible and allows different data formats to be transmitted. The formats allowed by the simulation tool written for this thesis are the Transport Stream, Asynchronous Transfer Mode (ATM), and Aggregated ATM Cell Encapsulation. It is important to be aware of the packet format used for transport to understand the underlying mechanisms affecting the received data. This section of the appendix presents a summary of the three above-mentioned formats.

A.1 Transport Stream

Transport Stream (TS) is a radical departure from previous video formatting standards. Fixed-length packet coding has several advantages over variable-length encoding, such as robustness and flexibility. However, the format's flexibility also yields a complex architecture. The following summary of the layout of the stream format provides a simplified glimpse of the capabilities of the standard.

A.1.1 Packet Format

Figure A.1 presents the format for each packet in the TS format. Each packet is 188 bytes long and is preceded by a sync byte (0x47).

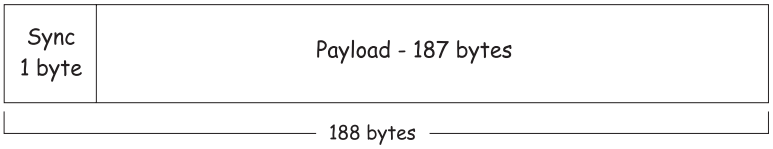

Figure A.1: Transport Stream Packet

The sync byte is inverted every eighth packet (0xB8). The inverted sync byte is used by the DAVIC communications system to reset the randomizer register.

A.1.2 Packet Header

Table A.1 describes the packet header, and the terms listed in Table A.1 are defined below. Each TS packet contains this header information; the information starts after the sync byte in the payload section of the packet.

Table A.1: Transport Stream Packet Header Structure

Description	Length (bits)
Error Indicator	1
Payload Unit Start	1
Transport Priority	1
PID	13
Scrambling Control	2
Adaptation Field Control	2
Continuity Counter	4
Adaptation Field (Optional)	-

Error Indicator - This is a single bit that signals if there is an error in the packet. The bit is set to '1' if an error is detected in the packet. Modules other than the MPEG encoder/decoder may change the value of this bit.

Payload Unit Start - This bit is set to '1' if packet contains the beginning of either Program Specific Information (PSI) or Packetized Elementary Stream (PES). PSI and PES packets will be discussed later in the description of the TS structure.

Transport Priority - A '1' means that this packet has higher priority than other packets with the same Packet IDentifier (PID).

PID - Table A.2 contains the list of possible Packet IDentifiers (PIDs).

Table A.2: PID Table

Value	Description
0x0000	Program Association Table
0x0001	Conditional Access Table
$0x0002 - 0x000F$	Reserved
$0x0010 - 0x1FFE$	Network_PID, Program_map_PID, elementary_PID, or other
$0x1FFF$	Null Packet

Scrambling Control - Defines the type of scrambling used. Value '00' means no scrambling. Values '01,' '10,' '11' have meaning that is user-defined.

Adaptation Field Control - The code for these two bits is defined by Table A.3

Table A.3: Adaptation Field Table

Value	Description
00	Reserved
01	No adaptation field, payload only
10	Adaptation field only, no payload
11	Adaptation field & payload

Continuity Counter - This counter increases with every packet with the same PID. The counter resets to zero when the counter reaches overflow. Increments on the counter are done only when the "Adaptation Field Control" has values '01' or '11.'

Adaptation Field - This is a variable length field that contains information used to deal with discontinuities, priority values for elementary streams, and information related to timing and clock referencing.

A.1.3 Program Specific Information

Program Specific Information (PSI) is information that is either needed by the decoder to demultiplex the incoming packets or is user-defined private information. Table A.4 presents a breakdown of the types of information that are classified as PSI.

Table A.4: Program Specific Information

Structure Name	Type	PID Number	Description
Program Association Table (PAT)	ISO-13818-1	0x00	Associates Program Number with PMT PID
Program Map Table (PMT)	ISO-13818-1	From PAT	Specifies PID in elementary streams for each program
Network Information Table (NIT)	User-Defined	From PAT	System information (channel number, level of error protection, etc.)
Conditional Access Table (CAT)	ISO-13818-1	0x01	Associates Entitlement Management Message with PID

All the types of information covered under PSI are important but for varying purposes. NIT is useful to define the system configuration during the system startup, and CAT is useful when there is secure data in the stream (such a premium channels and pay-per-view). These two formats (NIT and, to some extent, CAT) are important when the system is implemented and when service needs

to be expanded; they are not necessarily critical when attempting to understand the functionality of MPEG-2/TS.

PAT and PMT, on the other hand, are crucial parts of the transport stream format. They determine the association of individual programs in the streams as well as the association of different elementary streams within a single program.

A.1.4 Program Association Table

The Program Association Table (PAT) can be considered an electronic *TV Guide*. This table contains a list of all programs that are available in this particular stream. The list association is based on each program's thirteen-bit PID. The packets defining the PAT always have the PID set to 0x000. Table A.5 presents the PAT format.

The PAT is not necessarily one section (packet) long. The structure of the PAT is such that it allows a single table to be made up of several packets.

The program number list is described in Table A.6 and is of variable length because the number of programs that the Transport Stream can carry is variable. The number of programs that are carried by the PAT are determined by calculating the number of programs needed in the description to achieve the section length described in the "section_length" field of the PAT.

The values allowed for the Table_ID are listed in Table A.7.

The program information is structured on several layers. The top layer is the PAT, the table that contains a guide to all the programs that are available on the system. The basic unit used for controlling packet traffic is the PID, and the basic unit for program control within the PAT is the program number. The PAT (PID = 0x0000) contains a list of all the PIDs in the program and a list of all the program numbers. Under the PAT, the programs for that particular Transport Stream are listed with reference to program number. Each program number is associated with a particular PID that must be between 0x40 and 0xFE as seen in Table A.7.

The program with program number = 0 is the program with the network information, which is user-defined. The ISO-13818-1 standard does not define the contents of the network information

Table A.5: Program Association Table

Name	Number of Bits	Description
Table ID	8	Set to 0x00
section_syntax_indicator	1	Set to '1'
'0'	1	Single bit set to '0'
reserved	2	-
section_length	12	Value less than 1021 (0x3FD) specifying section length in bytes
transport_stream_id	16	User-defined value to identify this Transport Stream within the network
reserved	2	-
version_number	5	version number for the current PAT, resets to zero past maximum value
current_next_indicator	1	a '1' indicates this is the currently applicable PAT, '0' means the table is not yet applicable, but is the next one to become valid
section_number	8	starts at '0' and increments with every new section
last_section_number	8	last section number of complete PAT
program number list	Variable	see table A.6
CRC_32	32	CRC value that returns zero for the whole PAT

Table A.6: Program List Structure

Name	Number of Bits	Description
program_number	16	User-defined value that identifies the program for which the following information applies; if set to zero, the PID reference is the network_PID
reserved	3	-
network_PID	13	(If program_number == 0 only) specifies PID for transport packets with network information
program_map_ID	13	(If program_number != 0 only) specifies PID for transport packets with program_map_section applicable to the program with the specified program_number

Table A.7: Table_ID Values

Value	Description
0x00	program_association_table
0x01	conditional_access_section(CA_section)
0x02	TS_program_map_section
$0x03 - 0x3F$	ITU-T Rec. H.222.0-ISO/IEC 13818-1 reserved
$0x40 - 0xFE$	User private
$0xFF$	Forbidden

table (NIT). When used, the NIT contains the values needed to define the system, such as the channel used, bandwidth for the channel, modulation technique, and filter selection. If a program number in the PAT is not zero, it is a regular program (a collection of video and audio streams collected into a single program). All the available programs under the current Transport Stream are listed in the PAT, and an association is formed with their respective PIDs.

A.1.5 Program Map Table

Each program is headed by the Program Map Table (PMT). The PMT has the PID that was defined under the PAT as associated with this particular program. In short, the function of this table is to define the different elementary streams (video/audio) that will make up this program. The parameters for these streams and the whole program are also defined in the PMT. Table A.8 describes the structure of a Program Map Section (PMS). The PMT is comprised of several PMSs. Each PMS is in a single packet (section). Several PIDs can be used to describe a single PMT. The program_number is the common symbol that joins several streams with separate PIDs and is used to reconstruct the program.

The program descriptor(s) are structures that describe the program as a whole and are applicable to all streams. A list of descriptors available under ISO/IEC-13818-1 is seen in Table A.9.

The descriptor functions cover several program and elementary stream properties. However, given the limited nature of the system that is simulated, only three descriptors will be explored, the video_stream_descriptor, the target_background_grid_descriptor, and the maximum_bitrate_descriptor.

The reason for avoiding the other descriptors is as follows.

audio_stream_descriptor - This version of the simulation will use only video, so no audio handling functions are needed.

hierarchy_descriptor - This is used when different types of scalability are used. The simulation version discussed here will not use any type of scalability (Spatial, SNR, Temporal, or Data partitioning).

registration_descriptor - There is no private data in the format selected for the simulation.

Table A.8: Program Map Section

Syntax	Number of Bits	Description
Table ID	8	PID corresponding to the current program described as set in PAT
section_syntax_indicator	1	set to '1'
'0'	1	set to '0'
reserved	2	-
section_length	12	first two bits set to '00,' other 10 bits describe length of header in bytes
program_number	16	program number defined in PAT
reserved	2	-
version_number	5	version number of the current PMS
current_next_indicator	1	if set to '1,' the current PMS is applicable. If set to '0,' then the PMS is not yet applicable, but is the next one to become valid
section_number	8	set to 0x00
last_section_number	8	set to 0x00
reserved	3	-
PCR_PID	13	PID of packet with Program Clock Reference (PCR), if no PCR is defined, set field to 0x1FFF
reserved	4	-
program_info_length	12	the first two bits of this field are set to '00'; the next ten bits describe the number of bytes in the list of descriptors and stream descriptions
Program descriptor(s)	Variable	structures containing information applicable to the whole program
Stream(s) information	Variable	structure containing information on every stream and the descriptor(s) for all streams

Table A.9: Program and Elemental Stream Descriptors

Tag Value	Name	Description
2	video_stream_descriptor	Coding parameters for video elementary stream
3	audio_stream_descriptor	Coding parameters for audio elementary stream
4	hierarchy_descriptor	Identifies program elements with hierarchical-coded video & audio
5	registration_descriptor	Method to identify formats of private data
6	data_stream_alignment_descriptor	Describes the alignment type in the associated elementary stream
7	target_background_grid_descriptor	Used to describe an area of the screen with unit background (featureless) video
8	video_window_descriptor	Describes area where currently elementary stream is displayed
9	CA_descriptor	Conditional Access information for the decoder
10	ISO_639_language_descriptor	Identifies language(s) used on stream
11	system_clock_descriptor	Information on clock used to create timestamps
12	multiplex_buffer_utilization_descriptor	Provides bounds for decoder buffer size
13	copyright_descriptor	Allows identification of material ownership
14	maximum_bitrate_descriptor	Describes maximum bitrate (including overhead) of program or stream
15	private_data_indicator_descriptor	User-defined
16	smoothing_buffer_descriptor	Information on smoothing buffer used
17	STD_descriptor	Identifies type of buffering used in decoder (TS only)
18	IBP_descriptor	Describes structure of I-B-P frames

data_stream_alignment_descriptor - There is a single stream of video per program, so no inter-stream alignment functions are necessary.

video_window_descriptor - This descriptor describes the active window with respect to the background window. This particular simulation uses a single viewing area with a single stream, so this descriptor would be defaulted to zero. If different-sized pictures or picture-in-picture were used, this descriptor would be necessary; otherwise, it is redundant.

CA_descriptor - There are no conditional access streams in the simulated stream.

ISO_639_language_descriptor - This descriptor is used to describe the audio type used (e.g., hearing impaired), so it is not valid for this version.

system_clock_descriptor - The decoder clock is not an issue in this simulation. The decoder is too slow to decode the streams at the speed required, so an extra clock mechanism would further slow decoding performance without any gains.

multiplex_buffer_utilization_descriptor - Demultiplexing at the receiver is a post-processing operation in this simulation, making buffering and overflow non-issues.

copyright_descriptor - Copyright information on the stream would add an information overhead with no gains in performance, so this descriptor is not used.

private_data_indicator_descriptor - No private data is sent in this version of the simulation.

smoothing_buffer_descriptor - No smoothing buffer is used. Decoding is a post-processing operation.

STD_descriptor - Decoding is performed as post-processing, so no buffering information is needed.

IBP_descriptor - This descriptor describes the IBP format in each Group-of-Pictures (GOP). It seems redundant to include this information outside the actual elemental stream.

The three descriptors that will be discussed in detail fall under both categories, program descriptors and stream descriptors. Two descriptors, the target_background_stream_descriptor and the maximum_bitrate_descriptor, describe the parameters used for the whole program. The other descriptor, the video_stream_descriptor, falls under the category of stream descriptor.

The two program descriptors are included in the "Program descriptor(s)" section of the PMT. The

stream descriptor is included in a structure inside the "Stream(s) information" section of the PMT. The structure of the 'Stream(s) information' section is seen in Table A.10.

Table A.10: Stream Information Structure

Name	Number of Bits	Description
stream_type	8	Type of stream that is described in the structure
reserved	3	-
elementary_PID	13	PID of transport packets that will carry the described PID
reserved	4	-
ES_info_length	12	The first two bits are set to '00'; the other ten bits describe the length of the descriptor field(s) in number of bytes
descriptor(s)	Variable	Stream Descriptor Field(s)

The 'stream_type' field describes the type of elemental stream that is described in the structure. The list of stream types available is seen in Table A.11.

The structure described in Table A.10 is a variable-length structure. The decoder has three pieces of information that allow it to calculate the length of the stream information structure. First, the decoder knows the length of the whole descriptor list; it is given in the ES_info_length field. Second, the length of each descriptor field is included in the descriptor structure. Third, the descriptor tag is shown at the beginning of each descriptor structure.

The simulated system uses three descriptor structures. These structures are placed in their appropriate place in the PMT. The position for the descriptor field may be in either the program descriptor section or the stream descriptor section of the structure.

The target_background_stream_descriptor describes the viewing area that will be used in this program. The structure for this descriptor is seen in Table A.12.

The information for this descriptor can be obtained directly from the sequence being transported.

The maximum_bitrate_descriptor describes the maximum bitrate that the system is to expect in

Table A.11: Stream Type Table

Value	Description
0x00	Reserved
0x01	ISO/IEC-11172-2 video
0x02	ITU-T H.262—ISO/IEC-13818-2 video or ISO/IEC-11172-2 constrained parameter video
0x03	ISO/IEC-11172-3 audio
0x04	ISO/IEC-13818-3 audio
0x05	ITU-T H.222.0—ISO/IEC-13818-1 private_sections
0x06	ITU-T H.222.0—ISO/IEC-13818-1 PES packets with private data
0x07	ISO/IEC-13522-MHEG
0x08	Annex A - DSM CC
0x09	ITU-T H.222.1
0x0A	ISO/IEC-13818-6 type A
0x0B	ISO/IEC-13818-6 type B
0x0C	ISO/IEC-13818-6 type C
0x0D	ISO/IEC-13818-6 type D
0x0E	ISO/IEC-13818-1 auxiliary
0x0F − 0x7F	ITU-T H.222.0—ISO/IEC-13818-1 reserved
0x80 − 0xFF	User private

Table A.12: Target Background Stream Descriptor Structure

Name	Number of Bits	Description
descriptor_tag	8	Unique tag for this descriptor (0x07)
descriptor_length	8	Descriptor length, in bytes
horizontal_size	14	Size of horizontal background grid in pixels
vertical_size	14	Size of vertical background grid in pixels
aspect_ratio_information	4	Aspect ratio of display as described in ISO/IEC-13818-2

the transmission. The structure is described by Table A.13.

Table A.13: Maximum Bitrate Descriptor Structure

Name	Number of Bits	Description
descriptor_tag	8	Unique tag for this descriptor (0x07)
descriptor_length	8	Descriptor length in bytes
reserved	2	-
maximum_bitrate	22	Expressed in fifty bytes per second units
aspect_ratio_information	4	Aspect ratio of display as described in ISO/IEC-13818-2

The maximum bit-rate is described by a single twenty-two-bit integer. The unit of the integer is fifty bytes per second. The maximum bit-rate should be the maximum expected bit-rate for the whole transmission, including transport stream overhead.

The video_stream_descriptor describes a single elemental stream. The structure for the descriptor is seen in Table A.14.

The frame code is available from the ISO/IEC-13818-2 standard. If the multiple_frame_rate_flag is set, additional rates may be present on the stream. The allowed additional rates are listed on Table A.15.

Table A.14: Video Stream Descriptor Structure

Name	Number of Bits	Description
descriptor_tag	8	Unique tag for this descriptor (0x07)
descriptor_length	8	Descriptor length in bytes
multiple_frame_rate_flag	1	A '1' signifies that multiple frame rates may be present in the stream; '0' means that the entire stream is a single rate
frame_rate_code	4	Code representing sequence frame rate
MPEG_1_only_flag	1	If '1,' only ISO/IEC-11172-2 data, else ISO/IEC-13818-2 data
constrained_parameter_flag	1	If '1,' only constrained ISO/IEC-11172-2 data, else both constrained and unconstrained ISO/IEC-11172-2 data
still_picture_flag	1	A '1' signifies that there are only still pictures on the stream, else moving data may be added to the stream
MPEG_2_only	16	This field exists only if MPEG_1_only is set to '0'

Table A.15: Additional Frame Rates

Coded Value	Additional Rates
23,976	-
24,0	23,976
25,0	-
29,97	23,976
30,0	23,976 24,0 29,97
50,0	25,0
59,94	23,976 29,97
60,0	23,976 24,0 29,97 30,0 59,94

If the sequence is not MPEG-1 only, i.e., if the MPEG_1_only_flag is set to '0,' the MPEG_2_only field needs to be added to the structure. This field is shown in Table A.16.

Table A.16: MPEG-2 Only Field

Name	Number of Bits	Description
profile_and_level_indication	8	This value indicates a profile and level higher than or equal to any profile and level in the associated elementary stream
chroma_format	2	This field is higher than or equal to any chroma field in the associated elementary stream; all ISO/IEC-11172-2 streams have chroma '01' set to 4:2:0
frame_rate_extension_flag	1	A '1' signifies that any of the frame_rate_extension fields on the elementary stream are non-zero; all ISO/IEC-11172-2 streams are constrained with zero extension fields
reserved	5	-

Once the PMT is complete, the video is transmitted. The video packets are in Packetized Elementary Stream (PES).

A.1.6 Packetized Elementary Stream

PES provides a structure to pass timing information to the decoder. This timing information allows this particular elementary stream to be multiplexed into the whole program. The timing information is relevant to decoders that need to deal with several elementary streams and with hardware decoders that need to present the video data in real time. However, since the simulation needs to deal with only a single video stream, the PES can be greatly simplified.

The PES still needs the TS header (four bytes). These four bytes are the synchronization byte

(0x47 or 0x$B8$) followed by the three header bytes described in A.1.

The simplified PES header is made up of eight bytes as seen on Table A.17.

Table A.17: Packetized Elementary Stream Header

Name	Number of Bits	Description
packet_start_code_prefix	24	0x000001
stream_id	8	Stream identifier
PES_packet_length	16	Length of packet - allowed to be 0 (undefined) in MPEG-2 video elementary stream
10	2	'10'
PES_scrambling_control	2	No scrambling - '00'
PES_priority	1	High priority - '1'
data_alignment_indicator	1	Undefined alignment - '0'
copyright	1	No copyright - '0'
original_or_copy	1	Original - '1'
PTS_DTS_flag	2	No Presentation Time Stamp or Decoding Time Stamp - '00'
ESCR_flag	1	No Elementary Stream Clock Reference - '0'
ES_rate_flag	1	No Elementary Stream Rate in packet - '0'
DSM_trick_mode_flag	1	No tricks (Fast Forward, etc) - '0'
additional_copy_info	1	No additional information field - '0'
PES_CRC_flag	1	No CRC on this packet - '0'
PES_extension_flag	1	No PES extension information - '0'
PES_header_data_length	8	No header data, length = '0000'

The table provides information about the existence of extended field information fields. In the simplified form that PES is presented, these fields are not present, so the header is fairly empty. The timing information available in the PES header is provided by the PRS (Presentation Time Stamp) and the DTS (Decoding Time Stamp). These fields are left blank since this information is

not necessary in the decoder implemented in the simulation. All other fields present information whose value can be assumed (such as priority) or ignored (such as PES_priority). Some fields, like the CRC, are used only for network maintenance and are not intended for use by the decoder.

The valid values for the stream identifier are dependent on the type of elementary stream in the PES. For ISO-13818-2 or ISO-11172-2, the valid value for the field is 1110xxxx where xxxx is a unique video stream number.

The elementary video stream is then parsed after the header. If the end of the elementary stream does not match the TS packet size, the rest of the packet is padded with 0xFF-valued bytes.

A.2 Asynchronous Transfer Mode

The uplink in the system defined by DAVIC uses Asynchronous Transfer Mode (ATM) as its basic packet format. ATM is currently the standard packet format for high-speed asynchronous packet data.

A.2.1 ATM format

ATM packets are fifty-three bytes long. Each packet is made up of a five byte header followed by forty-eight bytes of payload. There are two formats for ATM, User-Network Interface (UNI) and Network-Network Interface (NNI). The header format for the NNI is described by Table A.18.

Table A.18: ATM Network-Network Interface

Name	Number of Bits
Virtual Path Identifier	12
Virtual Channel Identifier	16
Payload Type	3
Cell Loss Priority	1
Header Error Control	8

The *Virtual Path Identifier* (VPI) is a twelve-bit field that describes paths for the data to follow

within the network. This is a network-only feature independent of the end user.

The *Virtual Channel Identifier* (VCI), on the other hand, is a sixteen-bit field that describes the connection to the end-user. Unlike the VPI, the VCI is the direct access point of the user to the network. By contrast, the VPI provides routing information within the network for data transfer through the backbone.

The *Payload Type* is a three-bit field that describes the type of data that the ATM packet carries. Table A.19 shows the eight formats that are available within ATM.

Table A.19: Payload Type Formats

Code	Description
000	User data cell, no congestion, AAU=0
001	User data cell, no congestion, AAU=1
010	User data cell, congested, AAU=0
011	User data cell, congested, AAU=1
100	OAM F5 segment associated cell
101	OAM F5 end-to-end associated cell
110	Resource management cell
111	Reserved for future function

The first four fields denote user information, and the next three items denote network functionality. The AAU bit is used to pass information from end-user to end-user through the network.

OAM (Operation and Maintenance) is designed to do maintenance on both the physical layer and data-link layer of the network. Its main functions are: performance monitoring, defect and failure detection, system protection, failure or performance information, and fault localization. The table mentions F5, a term associated with the virtual channel of the network. It denotes the level of flow that is being addressed. The extra network management packets are used to do network maintenance transparently to the user.

The only difference between the NNI and the UNI is in the VPI. The header format for the UNI is described by Table A.20.

Table A.20: ATM User-Network Interface

Name	Number of Bits
Generic Flow Control	4
Virtual Path Identifier	8
Virtual Channel Identifier	16
Payload Type	3
Cell Loss Priority	1
Header Error Control	8

The *Generic Flow Control* (GFC) is available only for the UNI, so it is not transported over the network. It is used for user-control on the local data flow to control the quality of service at the user terminal. The packet format is bi-directional, so it can also be used to control flow at the network side of the link.

The VPI in the UNI is limited (there are only eight bits) compared to the format in the NNI packet header. The main reason for this difference is to offer greater flexibility to the network layer to manage the data paths within the network. When the UNI is handling the information, the extra four bits can be used for local flow control within the user's domain.

The forty-eight-byte payload directly follows the packet header in the ATM packet.

The DAVIC-compliant uplink uses ATM as its basic data transport format. The downlink can also carry ATM. The downlink AT packets must be packed in 188-byte format essentially encapsulating ATM in larger packets.

A.3 Aggregated ATM Cell Encapsulation

For the transport of ATM packets (fifty-three bytes) in MPEG-2/TS-sized packets, it is necessary to place a group of ATM packets into a structure that matches the larger transport format. The structure described by the DAVIC guidelines propose a method of aggregated ATM cell encapsulation, or stuffing. Figure A.2 shows the structure of the packets to accommodate ATM transmission.

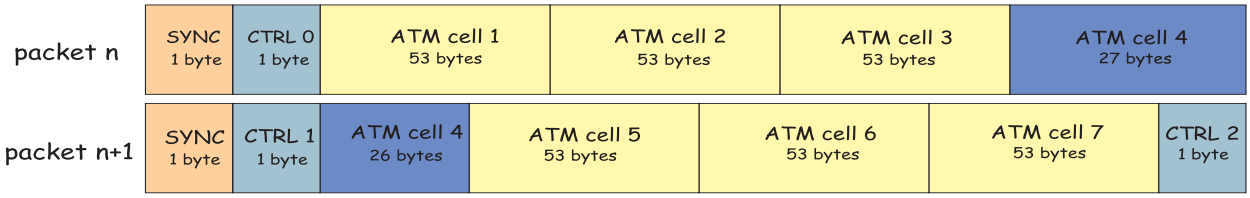

Figure A.2: Imbedded ATM Framing

The guidelines call for the ATM packets to be packed into pairs of 188-byte-long packets. Each pair of long packets carries seven ATM packets. If less than seven bytes of ATM data were available, the packets used to fill the balance would be null packets. The collection of ATM packets are converted into an aggregate group of cells that is then segmented into two pieces. The pieces are encapsulated into the larger packets. This form of encapsulation does not add a header format like MPEG. The packing and unpacking of ATM is performed at each side of the physical layer. In other words, the transmitter/receiver set has two sets of inputs, an MPEG set and an ATM set. The ATM packets are formed into larger packets for just the physical layer, and then they are disassembled at the receiver side of the link.

The control (CTRL) bytes that are added to the packet, apart from the synchronization byte and the aggregated ATM cells, follow the format seen in Table A.21, where E, P, and S are defined in Table A.22.

Table A.21: Control Byte Content

Name	Bit Layout	Description
CTRL0	E1PSSSSS	Indicates packet is first of set of two, first byte of ATM cell follows
CTRL1	E0PSSSSS	Indicates packet is second of set of two
CTRL2	Undefined	Reserved for Operation, Administration, and Maintenance (OAM)

The bit layout column describes each bit on the control byte. The CTRL2 byte is set for OAM and is system dependent.

The description of data packet generation given here for the different packaging methods is not complete for each method but presents all relevant information for the case under study. The packet format is crucial for the performance of the entire system. These areas include the study

Table A.22: Control Byte Codes

Code	Name	Description
E	Error	Set to '1' indicates at least one uncorrectable error in packet
P	Priority	A '1' means that the packet has greater priority over packets with priority '0'
S	Stuffing	Fixed set to '1'

of routing mechanisms, addressing problems, system maintenance, and fault detection. To better study the physical layer of the system, the problem was simplified, and several network issues were deemed beyond the scope of this study. These network issues, although relevant to the overall design, shed no light on the issue at hand, the relationship between video coding and channel coding in a wireless communication system. All the assumptions that were made concerning packet data generation are described in the chapter covering simulation.

Appendix B

Simulation Implementation

B.1 Architecture

The simulation is split into three parts, the MPEG encoder, the communications system, and the analysis tools. Although all parts are integrated into a single package, they can be treated as individual sections. The MPEG encoder/decoder set is composed of code that was originally downloaded from a University of California at Berkeley ftp site. This code was modified for robustness; it became necessary to add several safe-guards to ensure that the decoder did not crash when decoding a highly corrupted MPEG file. Part of this section is also the set of packetizing functions that were written to convert the MPEG video elementary stream to and from transport format. The communications system is composed of all transmitter/receiver modules and the channel model. The analysis tools are several routines that allow the quick assessment of PSNR, BER, λ_{error}, and $\lambda_{distance}$. All software written for this simulation is accessed through a unified user interface.

The first design decision to be made was the language/package that would be used. To appreciate this decision, consider a typical low bit-rate five-second video sequence. This video could have upwards of 6 or 7 Mbits. Added to the basic MPEG sequence, there would be channel coding (add anywhere between 10 and 100% overhead), taking the overall sequence to around 10 Mbits. Furthermore, the 10 Mbit sequence, after sampling (at baseband but sampling a sufficiently high number of samples per baseband bit, around ten) contains around 100 Msamples. This sequence

is very long, and it represents a single trial on the simulation. To keep the execution time to a reasonable interval, general-purpose modeling packages like MATLAB cannot be used for the computationally intensive iterative calculations involved in the simulation.

MATLAB is the simulation software of choice at the Mobile and Portable Radio Research Group (MPRG). It offers a large function library which allows a quick simulation development. An option to structured analysis packages is a low-level language like C. Whereas ANSI C has no standard communication-specific functions, making development time a slow and tedious process, it compiles efficient programs. A compromise was reached for the simulation. C would be used for anything iterative, while MATLAB would be used as much as possible for the simulation setup (calculation of filters and noise power) and any postprocessing (statistical analysis and graphing). This setup led to a reduced development time while maintaining a low execution time. MATLAB contains a family of C functions that allow a C program to use MATLAB as a slave process. The MATLAB C functions allow a quick and easy-to-implement transfer of information between the two processes.

This chapter will discuss the simulation structure. The discussion will be split into three sections: the MPEG viewer, the communications system, and the statistical tools. It is assumed that the reader has a basic understanding of the algorithms that were used, so there will be no detailed explanation of the structure of the underlying algorithm. The discussion will focus primarily on the design challenges that had to be addressed to complete the simulation.

B.2 MPEG Viewer

The MPEG-2 video standard was selected because it is the standard that the DAVIC guidelines mention as the motion standard of choice. There are two parts to the video section of the simulation, the encoder and the decoder. The encoder uses a set of frames to generate an MPEG sequence. The decoder uses that sequence to generate either a set of raw frames (like the ones the encoder uses to generate an MPEG file) or the MPEG video sequence seen in a viewable area on the computer screen.

An MPEG encoder/decoder set was found on an ftp site at the University of California at Berkeley. This stand-alone software allows one to convert a series of frames into an MPEG sequence. The

interface for the program is fairly cumbersome, but it works and it was available free from the net. Since it is already a stand-alone application and since there is no graphical interface in the program, it was left as a stand-alone application. The encoding/decoding process cannot be done real-time on a PC using this software, but it can be used to prepare MPEG sequences for the simulation.

An MPEG viewer was also found at the University of California at Berkeley. This viewer was available with its C source code. Unfortunately, the program was written for a Unix platform. A patch was found on a site at the Brigham Young University that allows the viewer to be ported to the PC. There were a few bugs in the patch. Once these bugs were cleared out, the program allowed MPEG-2 video elementary streams to be viewed on the PC.

The viewer was written using Microsoft Visual C. Instead of using C++ for the windowing code, runtime functions for windowing are used to generate the user interface and present the movie frames. This core interface is the interface that was used for the whole simulation.

B.2.1 Viewer Modifications

Several bugs were found with the viewer. These bugs dealt primarily with memory bound problems. The original program was not robust enough to deal with severely corrupted MPEG streams (such as those expected on a low-power wireless link). Several bounds and sanity checks had to be placed inside the viewer so that it could display a corrupted sequence without crashing.

The basic viewer contained 4 functions: file open, exit, play, and stop. These functions had to be extended considerably to allow the work that was planned.

The 'File' section of the program was edited, and the final version looks like Figure B.1. Several entries are seen on the menu. The original entries were the first and last entries on the menu. All the other entries deal with functionality that was added to deal with the different data format translations needed.

The "Control" section of the program was also edited seen in Figure B.2. Two entries were added, the "loop" section and the "Custom Control" section.

The 'loop' allows the sequence to be repeated indefinitely. The "Custom Control" section leads to

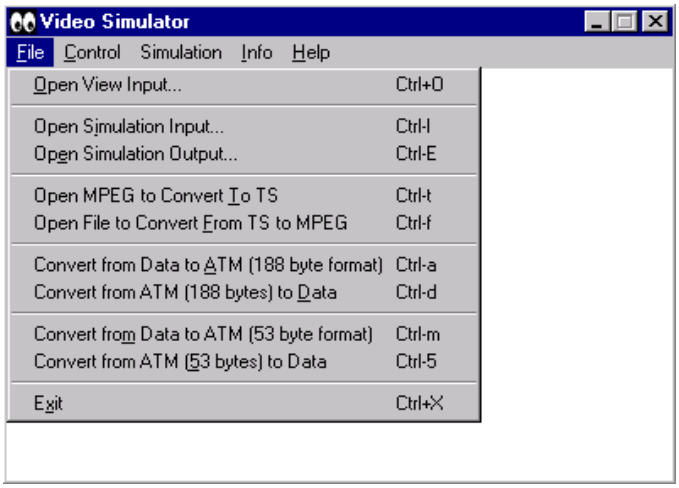

Figure B.1: File Options Screenshot

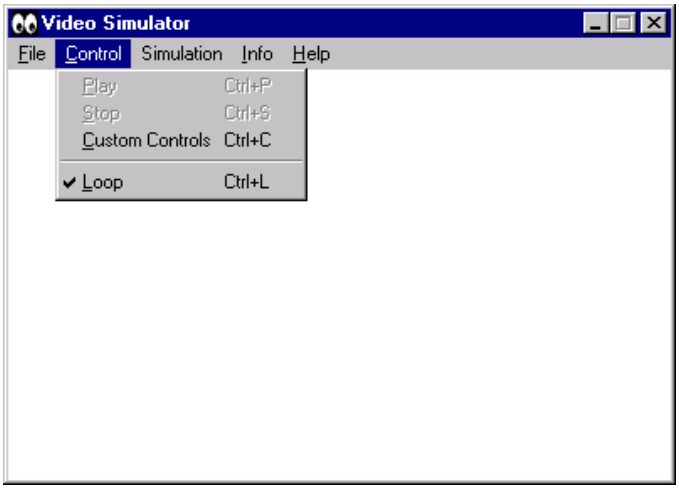

Figure B.2: Control Options Screenshot

the menu seen in Figure B.3. This menu option was added to allow more control over the display of the sequence. Speed control, pause, and single step allow considerably more control over the sequence than the old play and stop functions. Also, the frame counter gives the user instant feedback reference to the current position of the sequence.

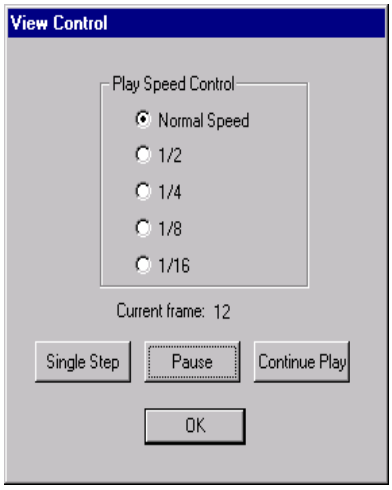

Figure B.3: Control Options Screenshot

Other additions were made to the interface to integrate the communication system simulation into the viewer and to include statistical analysis tools as well. These additions will be discussed in later sections of the chapter.

The data format input into the viewer is MPEG-2 elementary video stream, which is not the format that is transmitted on the channel. The simulation supports several formats: ATM, MPEG-2/TS, and Aggregated ATM Cell Encapsulation. A pre-processor that converts these formats to and from MPEG-2 elementary stream was written.

B.2.2 Data Formats

It is necessary to represent the data for transmission in different formats: MPEG-2/Transport Stream, ATM, and Aggregated ATM Cell Encapsulation. There are essentially two approaches possible for the system to handle these formats. The first approach is to write a basic packaging algorithm. This algorithm would package any data set into the appropriate format. The system

would then simulate the transmission. The resulting packaged format could then be taken by the viewer and internally parsed and viewed. This approach would require extensive modifications within the viewer.

The other approach to dealing with the format issue is to create a pre- and post-processor to handle the different formats. Each processor would be an independent application whose only connection to the communications simulation and MPEG viewer would be the common interface. Each processor receives a raw data file that is packaged according to the particular format, and the resulting file is output. The post-processor can then be used to convert the formatted file into the original raw format. This process, although somewhat cumbersome, is modular in design. This means that the complexity of each format is completely isolated within a single module. The viewer can be used to view the raw file before and after the full format processing is done.

I chose the latter form to handle the different formats that need to be addressed by the simulation. As seen in Figure B.1, each format, pre- and post-processor, has a single entry in the file menu option. To process a file, the format is selected on the menu, and a Windows standard file menu opens up. When the selected file is opened, the processor is automatically run and a file is created with the formatted output. There are only two possible outcomes, success and failure. The result of the operation is known by looking at the output file. If the output file exists and has a non-zero size, then the process was a success, else it failed. Since the processor is designed to handle files with many errors, it is robust. There is no error feedback to the user, but the processor cannot crash.

The actual processing is done depending on the format selected. Each format applies the rules that were presented in the background chapter of the paper. Since the language used is C, it was fairly simple to allocate the space necessary to process the information and to do the bit manipulation necessary to output the correct format.

The format of the file conversion algorithm testing was fairly tedious. Since there is no off-the-shelf viewer available on the net to test these formats, the format setting had to be tested by hand. A binary editor was used to look at the file that was created. The file's header was checked for content in the beginning of the file and at random points down the file (since each randomly-selected point is going to be some number of packets away, the relative position of the packet header was easy

to find). The file content matched the expected packet content. Later in the development of the simulation, the content of the packet was checked when it entered the communication system, and the resulting packet content matched the expected results. As a further check on the format, the reconstructed data files were checked using a binary error counter, and no bit errors were found.

B.3 Communication System

Whereas source code for the MPEG viewer was found on publicly-available sites license-free, source code to simulate the communications system was not available. All C code written in this project for the communications system and channel modeling was original code. The simulation presented several implementation challenges to be overcome.

B.3.1 Simulation Architecture

Several challenges came up in the original design of the communication system simulation. Since there are at least two basic systems to be modeled (uplink and downlink) and since combinations of both architectures would be useful to study particular events, it became necessary to create the simulation with as flexible a module layout as possible.

Figure B.4 describes the simulation layout. The program is split into several modules that are, for the most part, independent. As can be seen in the flow diagram, the input and output to each module is controlled by a set of generic pointers. Each module, if it is supposed to execute, will direct the pointers to its own local pointer structure, where the module executes its assigned task. Once the task is complete, the output data is assigned to the same generic pointer structure that was used as its input.

This structure may seem unnecessarily cumbersome, but it presents several advantages. The structure of the simulation is unknown apriori, which means that the overall structure has to be flexible enough to allow quick addition and removal of simulation modules. If all the modules point to the same structure for their input and output, the same function set can be used to handle inter-module communications, reducing inter-module dependencies.

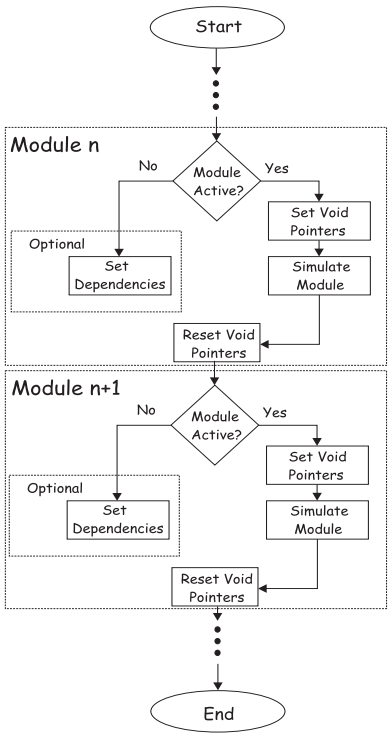

Figure B.4: Simulation Architecture

In practice, this concept proved to be considerably more complicated than Figure B.4 implies. Since C requires all memory to be allocated and since the size of the input vector will sometimes be different from the size of the output vector, dependencies arose that added a level of complexity to the program.

A setup step became necessary where the layout of the simulation would be prepared for the iterative process. This setup step has essentially the same structure as seen in Figure B.4. If the module is active, the size of the input and output vectors is calculated and the proper memory is allocated. Since the generic pointer never allocates anything, each module will have its own memory pre-allocated with a local structure keeping track of each memory set. In the setup step, a generic vector length is passed between setup functions. This means that the size of the output vector is calculated by a module based on some generic vector length that is passed to its input. Once the new vector length is calculated, the appropriate memory buffer(s) are allocated, and the module output vector length is passed to the generic vector length variable. Using this structure, the memory buffer(s) can be allocated by each module dynamically to fit the needs of the current simulation structure.

The nature of the data changes when it is modulated. Before modulation, the data can be represented with integer variables. Upon modulation, the data changes in structure. Instead of discrete pieces of data representing each number, data is represented by an I and Q data set. Matters are further complicated by the filtering algorithm where the I and Q data is sampled and passed through the pulse shaping filters. The first set of data, the discrete ones and zeros, can be represented by using integer type in C. The I and Q data must be represented by floating point numbers.

Although the generic pointer will be able to handle anything, the modules effecting the transition may change (different types of modulation or the use of the convolutional encoder). At this point, the "Optional" box in Figure B.4 plays a major role. This box is designed to handle the special cases where dependencies like the one just described appear. These dependencies are module-specific, and are there to handle the cases where the generic modular structure fails.

The special case handling only occurs around the convolutional encoder module. The convolutional encoder performs the modulation for the system (QPSK only). When the convolutional encoder is not available on the system or another type of modulation is used, there needs to be a function

that will perform the modulation for the module and inform the next module of what array size to expect. Also, this alternate function set needs to perform the data conversion for the absent module from integer to floating point.

Figure B.5 presents the simulation's user interface to select the layout of the simulation structure.

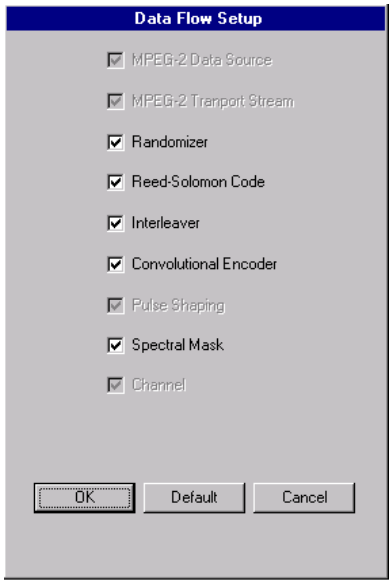

Figure B.5: Flow Control User Interface

The user has the option of turning modules on and off at the press of a button. Some modules, such as pulse shaping, are considered critical to the simulation and their exclusion is not an option.

B.3.2 Channel Modeling

When simulating a wireless communications system, care has to be taken to properly model the channel characteristics. This simulation must correctly model two effects to simulate the conditions of a terrestrial LMDS system. These conditions are the background thermal noise and vegetative fading, modeled as either lognormal fading or weak Rician fading. Rayleigh faiding is common in several wireless environments, and it was also included. Although the Rayleigh model does not apply to the LMDS channel, it has some peculiar characteristics that could make the channel's relevance to video coding quite significant.

The heart of any channel model is a good number generator. The number generator is the seed upon which the channel model will be based. If the seed possesses the wrong statistical characteristics, then the whole model will be inaccurate.

The common denominator for all channel models that are used in this simulation is the uniform random variable. The uniformly distributed random variable is used as the basis for the Gaussian random variable, which in turn is used to model all other random variables used in this simulation.

The number generator used for the uniform random variable is the Wichmann-Hill algorithm. This algorithm, discussed in the background chapter, generates a number sequence whose period is very long (in excess of 10^{12} samples). The number generator is based directly on Equations C.2 through C.5. Using the Box-Mueller algorithm, Equations C.6 & C.7, the uniform random variables are converted into independent Gaussian random variables.

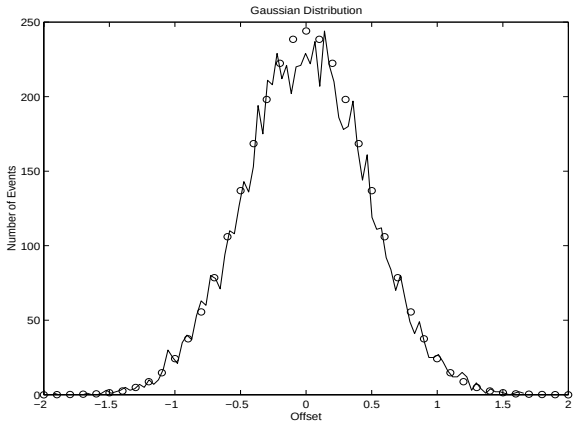

Figure B.6: Probability Distribution Function of Gaussian Random Variables

Figure B.6 shows the distribution achieved from the Wichmann-Hill/Box-Mueller combination. As can be seen from the graph, the achieved distribution closely matches the theoretical Gaussian curve.

Figure B.7 shows a scatter diagram of the Gaussian random variables that were generated using the Wichmann-Hill/Box-Mueller written in C. No discernible pattern is seen within the scatter plot, leading to the conclusion that the two generated random variables are not correlated. If some correlation existed between the variables, it would show up as a discernible pattern in the graph.

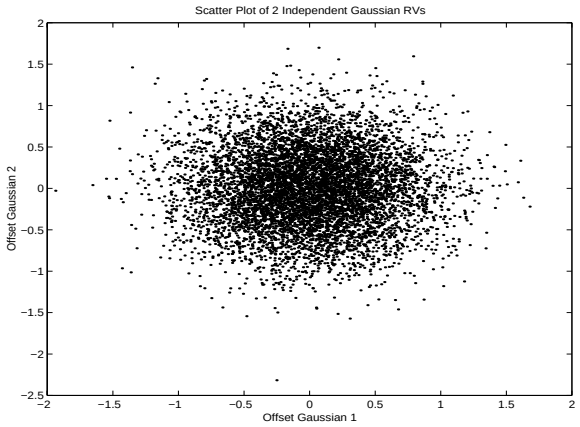

Figure B.7: Scatter Diagram of Two Gaussian Random Variables

The Rayleigh fading was done using Clarke's model. The resulting envelope is seen in Figure B.8.

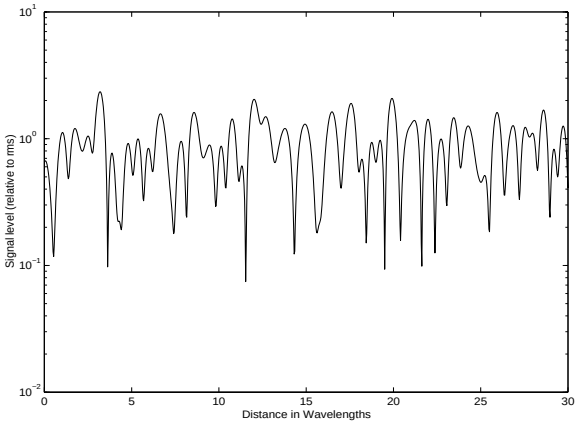

Figure B.8: Rayleigh Fading Envelope

The Rician and Lognormal random variables are offshoots of the Rayleigh and Gaussian distributions, respectively, and their simulation was straightforward.

B.3.3 Module Description

Each module presented a different set of challenges to be solved. It is assumed that the reader is familiar with the basic architecture of the modules, and although a general overview will be provided, the algorithms will not be discussed in great detail.

Data Acquisition

The simulation does not run on real time, so data acquistion is not a complicated affair. The simulation is given a basic packet size (Figure B.9).

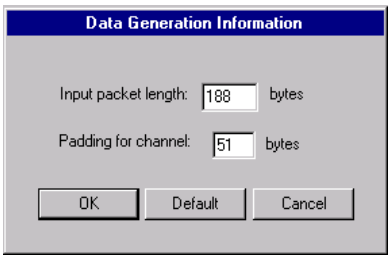

Figure B.9: Packet Generation User Interface

The dialog box requests from the user a basic packet size and the amount of padding that needs to be removed from the packet prior to modulation. This padding is zero padding that is added by the Reed-Solomon encoder to fit the structure that it is given. To save bandwidth, this padding is removed prior to modulation. The simulation will read a length of information defined by packet size from the file every time it iterates. The file needs to make an exact fit on the packet size. In other words, if the packet size is 100 bytes, then the file needs to be size $n \times 100$ where n is the number of packets sent (not known apriori). If the number of bytes does not match, an error will occur and the simulation exits. If the sizes do not match, chances are that there will be an underlying problem with the conversion algorithm or the simulation settings and under these circumnstances, the simulation results are probably not valid.

The data generation algorithm was tested using two methods. First, the data generation and data output functions were placed back-to-back with no other modules active in the system. The simulation was run, and the content of the buffer containing the data was checked. The binary contents should match the expected header information and flags. If the contents matched, the synchronization was assumed correct. The program was let run with a correct file. The resulting file was compared to the original file, and they matched perfectly, so the operation for a correctly sized input was verified. An improperly-sized file was then used, and the program exited without warning as expected. After these tests, the data generation algorithm was assumed to work correctly.

Randomizer

The randomizer is a Linear Feedback Shift Register (LFSR). The simulation's randomizer user interface is seen in Figure B.10

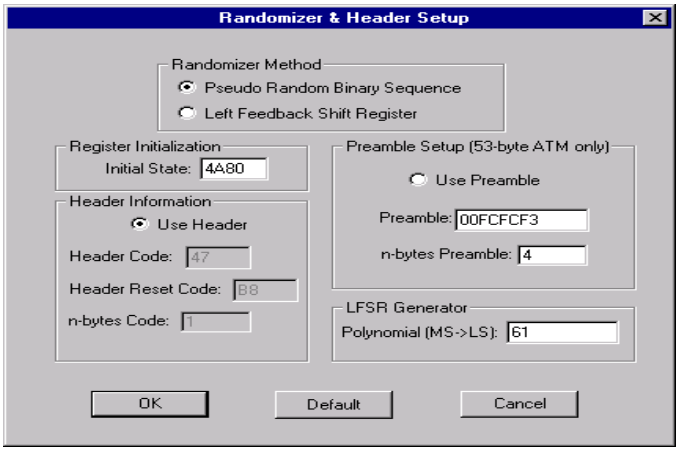

Figure B.10: Randomizer User Interface

The randomizer user interface lists two possible randomizer structures, an LFSR and a Pseudo Random Binary Sequence (PRBS). They both have the same underlying structure, an LFSR. The difference in nomenclature in the simulation is due to the same discrepancy in the DAVIC guidelines. The only difference between the two structures, the LFSR can be given any generator polynomial and the PRBS is hard-coded to a generator polynomial of $1 + x^{14} + x^{15}$ with a starting binary sequence of 100101010000000.

The randomizer was implemented in C. It follows a straightforward implementation approach, and no challenges arose in the generation of this module.

The derandomizer on the receiver is exactly equal to the randomizer, including the original state. Since the randomizer's output is a series of ones and zeros, and since that output is Xored with the data, Xoring the incoming data with the same set of ones and zeros will result in a complete reversal of the randomizer. One or zero Xored with the same value twice will yield the original value regardless of what the original value was.

The randomizer was tested using two methods. The randomizer and derandomizer were placed back-to-back. The output of the randomizer was calculated by hand for several cycles of the buffer

set. The computer's output matched the expected results. Also, the randomizer-derandomizer back-to-back set completely negated each other for the whole set, further validating the program function.

Reed-Solomon

The Reed-Solomon (RS) block code comes from the family of BCH codes. The RS code is a non-binary code; the alphabet of symbols that the code is designed to work on is not necessarily limited to just one and zero. The symbol list for the RS code is defined by a Galois field. Figure B.11 displays the user interface for the RS module in the simulation.

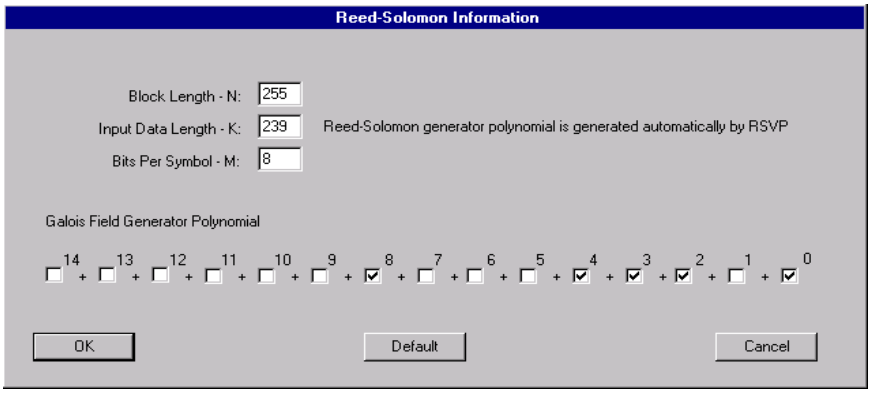

Figure B.11: Reed-Solomon User Interface

The only parameters needed to generate the full code are N (the length of the total code word), K (the length of the data part of the code word), M (the number of bits per symbol), and the generator polynomial for the galois field. The number of symbol errors that can be corrected by the RS code is t where $t = \frac{N-K}{2}$. The number of symbols that can be detected by the RS code is $2t$.

The Galois field is a finite number field. This field is generated based on a primitive, α. Each word in the whole number set is a power of α. In other words, the whole number set is described by the numbers α^0 through α^{2^N-1}. If $\alpha = 2$, the first M elements are defined by the primitive shifted to the right by $M-1$ (1, 2, 4, 8, 16, 32, ... , 2^{M-1}). The M^{th} word is defined by the Galois polynomial, or in the case of GF(256) where the polynomial is $x^8 + x^4 + x^3 + x^2 + 1$, the M^{th}

element is $x^4 + x^3 + x^2 + 1$, or 29. Every consecutive element after that is calculated by multiplying the previous element by α and Xoring the result with the M^{th} element if there is an overflow from the previous multiplication.

To handle the primitive numbers, all mathematics were performed by using the primitive's exponential instead of the actual number. The exponential was used as the index to a lookup table. The use of the exponential greatly simplifies the algorithm that must be performed every time there is an operation. For example, multiplication is just the addition of exponents. Since this is a finite field, an exponent value greater than the maximum value is the same as the exponential minus the maximum exponential value minus one. Addition, on the other hand, is just the Xoring of the actual values taken from the lookup table. A second lookup table, with the index being the primitive-derived value and the content of the register being the primitive exponential, was constructed to allow quick lookup of exponentials resulting from additions.

The actual code generation is accomplished dividing the data word by the RS generator function. Since the code is systematic, the resulting code word is the original data and the remainder of the division operation is appended to complete the full code word. The division is accomplished by using the logical circuit seen in Figure B.12 [23].

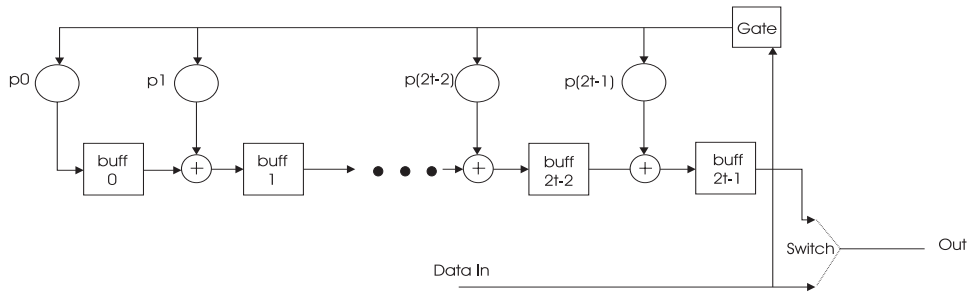

Figure B.12: Reed-Solomon Encoder

The operation of the circuit is fairly simple. At first, the gate is closed and the switch is connected to the data line. The data enters the encoder, sets the states of the buffers in the encoders, and at the same time exits the encoder. When all the data has passed, the gate is opened, and the switch is now connected to the buffer output. The buffer contents are then systematically shifted until they are all empty. The resulting code word will be $N + 2t$ elements long. Implementation

of the encoding algorithm was fairly straighforward. The biggest challenge in writing the encoder and galois field functions was in making sure that the galois field functions (primarily add and multiply) worked correctly.

Typical to most communication systems, the decoding process of a RS code is considerably more complicated than the encoder. There are several algorithms that can be used to decode a RS code. The decoder that I selected to implement is the Berlekamp-Massey algorithm. The detailed explanation of the decoding algorithm is beyond the scope of this thesis. In essence, the decoder works by generating a polynomial as the codeword enters the decoder. The generated polynomial's order is the number of symbol errors the code word has (order t at most). If the roots of the polynomial are not distinct, then the code failed. If the roots are unique, an error polynomial can be constructed. When this polynomial is added to the original code word, the resulting code word is the corrected word. For an detailed explanation of the Berlekamp-Massey algorithm, see [24]. Figure B.13 presents the error-correction process of the simulated RS decoder.

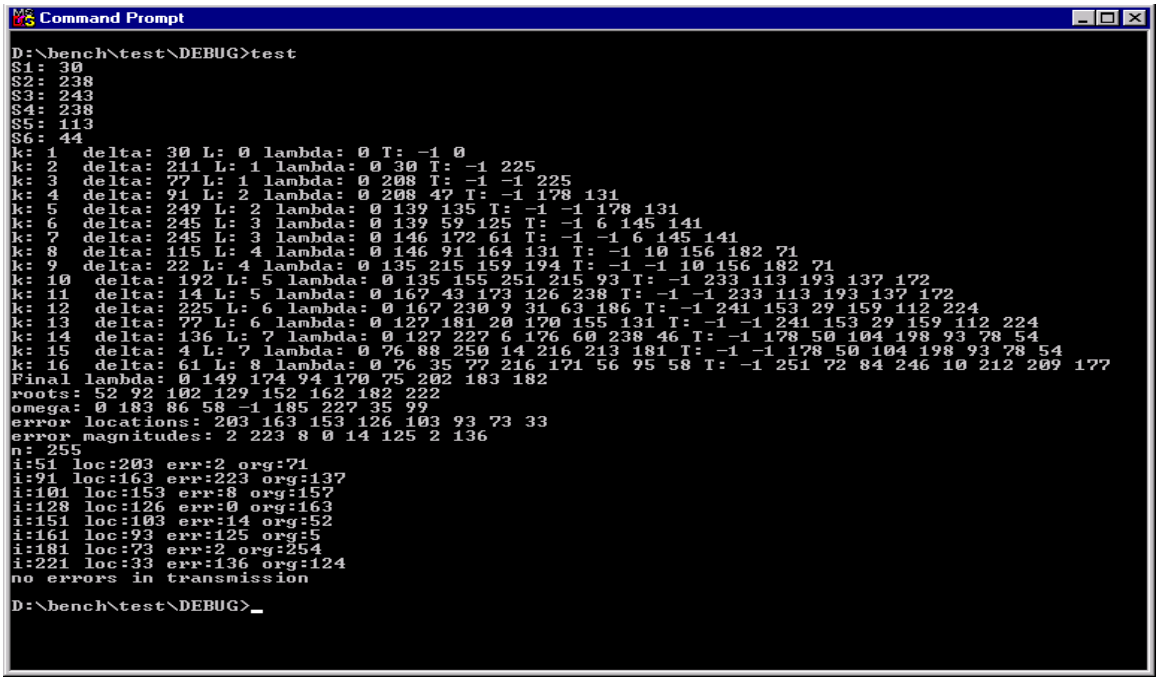

Figure B.13: Reed-Solomon Decoder

As can be seen on the screen output, the syndromes are calculated and the lambda polynomial is generated as the code word enters the decoder. At the end of the algorithm, the error polynomial

is generated and added to the original polynomial. The resulting polynomial is error-free.

Validation for the Reed-Solomon code was performed in two steps. Since the words can be very long, a simple RS code was implemented (the simulation is designed to accept different block sizes), and the RS coded word and decoded results were compared to the return value of the MATLAB RS encoder/decoder. The results for code success and failure were compared, and the results matched. The next step involved obtaining BER curves for the performance of the RS code. The curves are presented in the simulation performance section of the paper.

The main challenge in creating the Reed-Solomon decoder arose in the representation of polynomials. The solution that was used was to use a pointer to define each polynomial. A pointer was dynamically allocated a certain amount of memory, each memory slot being a single coefficient. Since all polynomials begin at the zeroth element, polynomial addition became a matter of making sure that the memory was allocated and padded on the most significant side of the polynomial to avoid the summation of values from un-allocated memory areas (which would otherwise lead to unpredictable results or a program crash).

Interleaver

The only information that is needed by the simulation to create the interleaver is the interleaver depth. Figure B.14 shows the simulation's interleaver input.

Figure B.14: Interleaver User Interface

The interleaver is implemented using a very simple algorithm. The incoming bits need to be mapped into an output buffer the same size as the original buffer. The mapping that is done depends only on the depth of the interleaver. The deinterleaver follows the same process as the interleaver only backwards.

Testing of the interleaver was done in two ways. The interleaver output indices were calculated by hand; the interleaver should place the data in the correct offset position of the output buffer. Once this relative position matched the expected results, the interleaver and deinterleaver were placed back-to-back. The resulting data set was compared to the original data set and no errors were found.

Convolutional Encoder

The convolutional encoder is a fairly simple algorithm. An example of a convolutional encoder is seen in Figure 2.3.

The convolutional encoder simulation requires considerably more setup information that the other modules in the system. Figure B.15 presents the user interface that acquires all necessary information to set up the convolutional encoder and the Viterbi algorithm.

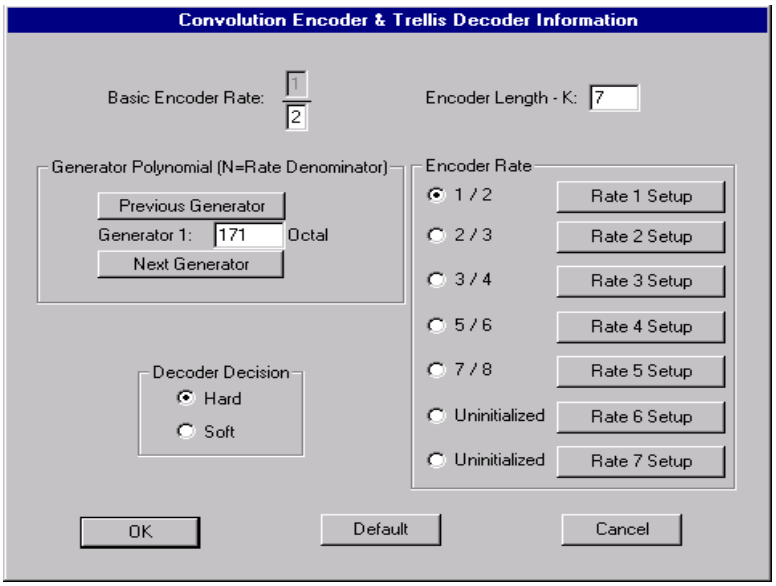

Figure B.15: Convolutional Encoder and Viterbi Decoder User Interface

The convolutional encoder screen can setup all information pertaining to how the convolutional encoder should work. Most of the information on the screen is self-explanatory. The only part that may leave some questions is the variable rate. The variable rate is achieved through the use of a puncture map. The puncture map for each rate can be set by pressing the "Rate n Setup" button.

When this button is pressed, the screen seen in Figure B.16 appears.

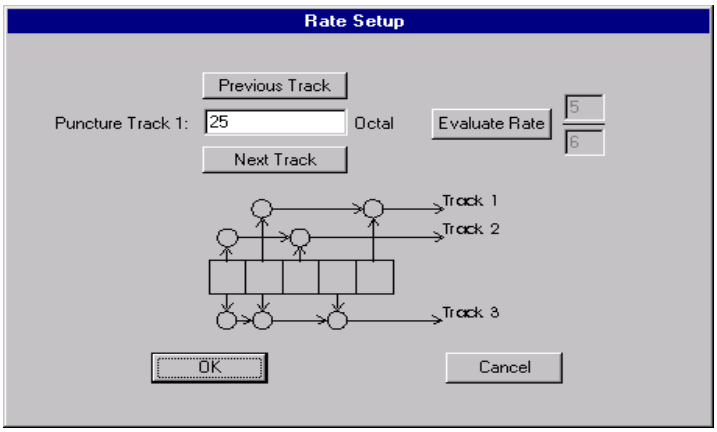

Figure B.16: Convolutional Encoder Rate Setup User Interface

This screen will setup the puncture map for each output track of the convolutional encoder. Once the puncture map has been set for each track, hitting "Evaluate Rate" will calculate the data rate of this puncture map. The mother dialog box will be altered with the new encoder rate when the child dialog box shuts down.

Modulation

Three modulation techniques were implemented in the simulator; QPSK, 16-QAM, and DQPSK. The constellations used for each of these formats are the ones described in the background chapter. Figures B.17, B.18, and B.19 show the eye diagram and the state transition for QPSK, 16-QAM, and DQPSK respectively. The modulated impulses are pulse-shaped using a root-raised-cosine filter.

Figures B.17 and B.19 are identical because even though each of the modulation schemes performs a transition for a different reason than the other scheme (one is absolutely mapped, the other is differentially mapped), the destinations for both modulation schemes are exactly the same.

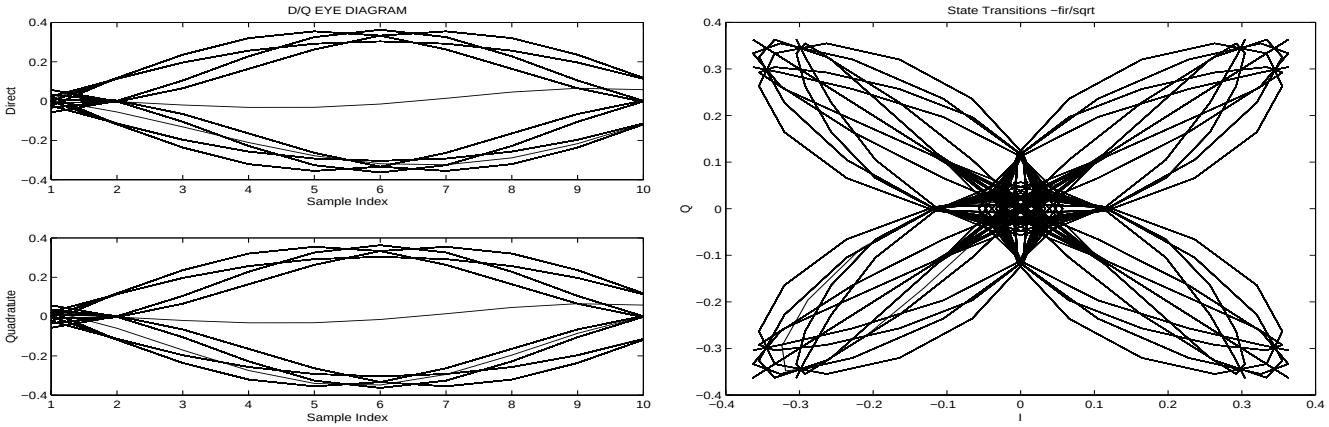

Figure B.17: QPSK Eye Diagram and State Transitions

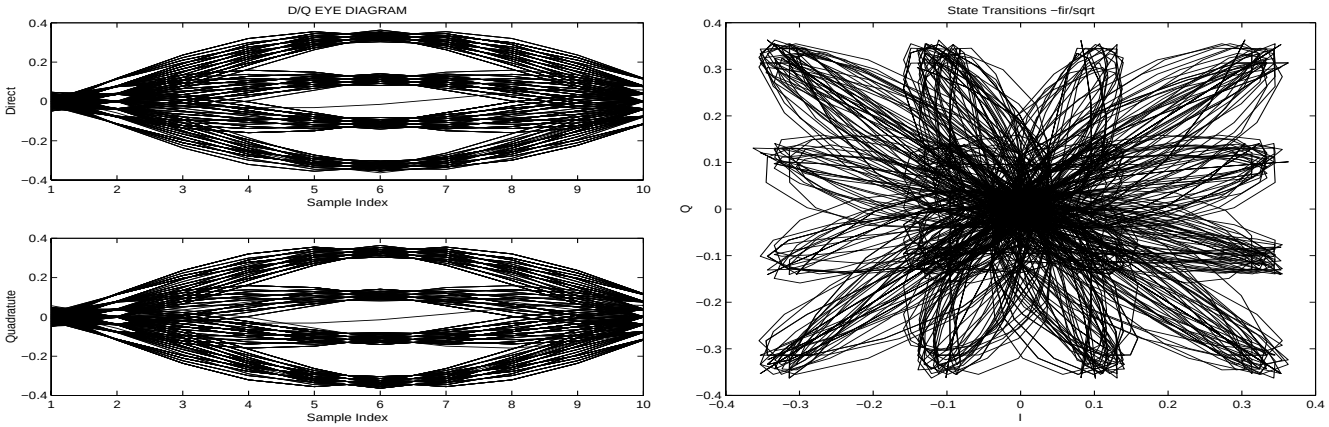

Figure B.18: 16-QAM Eye Diagram and State Transitions

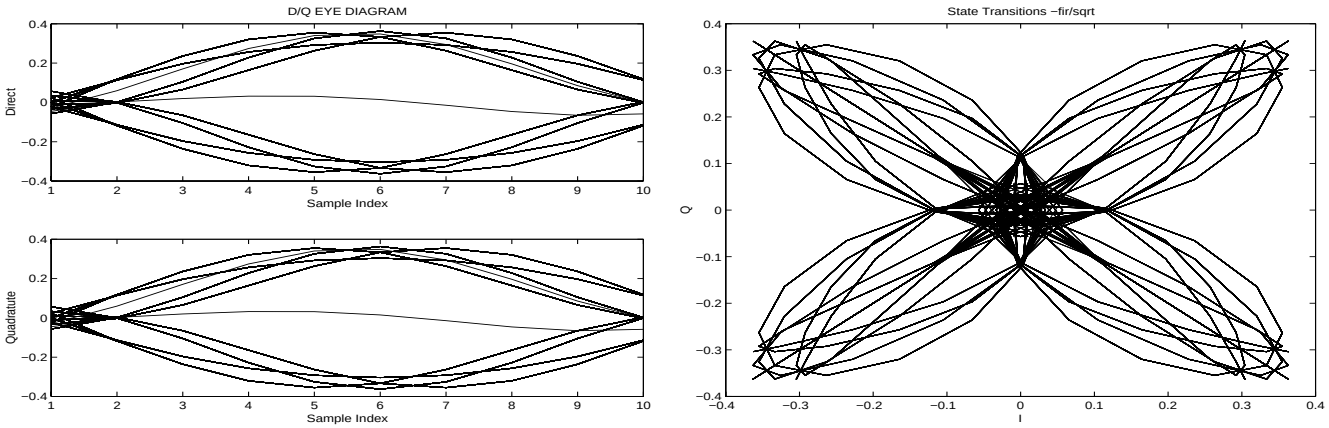

Figure B.19: DQPSK Eye Diagram and State Transitions

Pulse Shaping

The DAVIC standard calls for the modulated impulses to be pulse-shaped using a root-raised-cosine filter. The pulse-shaping was performed at run-time using a transposed-direct-form-II FIR filter, as seen in figure B.20 [25].

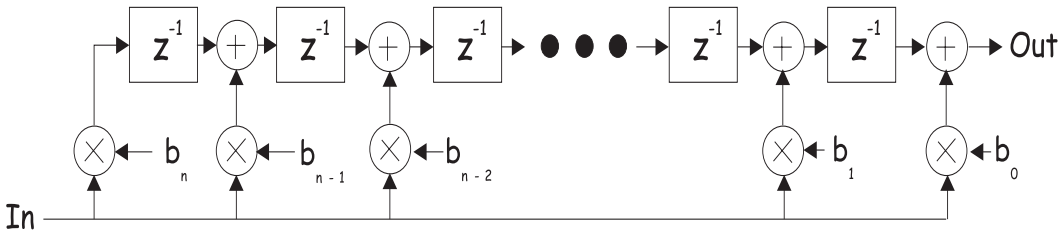

Figure B.20: Transposed Direct-Form II FIR Filter

Since the filter coefficients need to be designed only once, the MATLAB rcosine() m-file was used in the simulation initialization to generate the filter coefficients. MATLAB offers several Dynamic-Link-Libraries (DLLs), which can be linked to a C program at runtime. These DLLs offer the C program the opportunity to pass variables to MATLAB, execute a set of commands in MATLAB, and return a result to the C environment.

B.3.4 Calibration

Although generating a noise channel in a simulation is a straightforward process, calibrating the noise level is not. [18] presents equations B.1 through B.4 to determine the proper standard deviation for the Gaussian noise generators to represent a particular $\frac{E_b}{N_o}$.

E_b and N_o are defined by Equations B.1 and B.3, respectively.

$$E_b = \frac{P_{out}T}{k} \tag{B.1}$$

where P_{out}, defined by Equation B.2, is the average signal power, T is the symbol period, and k is the number of bits per symbol.

$$P_{out} = \frac{1}{2M} \sum_{n=1}^{M} |S_n|^2 \tag{B.2}$$

$$N_o = \frac{|H(0)|^2 B_N}{\sigma^2} \tag{B.3}$$

where $|H(0)|$ is the DC response of the receiver filters, B_N, defined by Equation B.4, is the noise bandwidth of the receiver filters, and σ^2 is the variance of the gaussian noise.

$$B_N = \frac{\int_{-\infty}^{\infty} |h(t)|^2 \, dt}{\left| \int_{-\infty}^{\infty} h(t) dt \right|^2} \tag{B.4}$$

To prove the system is calibrated correctly, it is important to compare the simulation results to the expected system performance based on analytical estimates or bounds.

Appendix C

Random Number Generation

It is important to generate random variables for the channel simulation such that they statistically represent the expected channel performance. A computer is, by definition, a deterministic machine, causing a basic problem. A computer knows only how to add and multiply. To that end, it is not possible to generate a sequence that is random. However, it is possible to generate a sequence that has the correct properties such that it looks as close to random as a deterministic approach can converge to. The main goal of a number generator is to generate a white sequence, a sequence whose spectral components are even over all frequencies. Another way of looking at this type of sequence is as a delta-correlated sequence. There are two types of white sequences that are particularly useful, the uniform-distributed random variable and the random binary sequence. Though both sequences are white, their basic distributions are different. The uniform distribution presents an even distribution across a number space. A binary sequence is a sequence whose entire number space is composed of two points, '1' and '0.' The uniform random variable is used to create other desired random variables, like Gaussian or Rayleigh distributions. Since data that is transmitted should be as uncorrelated as possible (correlated data can be predicted), it is important to create binary sequences that are as close to random as possible.

In the following sections, the generation of uniform random variables and binary sequences will be discussed. The two generation sections will be followed by a discussion of random variable transformation.

C.1 Uniform Random Variables

A uniform random variable can be created using a generator that follows the pattern seen in Equation C.1 [18]

$$X_{N+1} = (A \times X_N) \text{ Mod } (M) \text{ where all } X_N \text{ are relatively prime to } M. \tag{C.1}$$

The problem with this arrangement for a number generator is the sequence length. There can be no sequence that is longer than M elements. In other words, the number sequence may at most cover the full sequence from 0 to $M - 1$. To avoid periodicity, the random sequence needs to be longer than the data sequence on which noise will be applied. However, if the sequence is too long, Equation C.1 will tend to overflow the computer's buffers.

The Wichmann-Hill algorithm avoids this problem by using a series of equations to calculate a single uniform random variable [18]. Equations C.2-C.5 describe the Wichmann-Hill algorithm.

$$X_{N+1} = (171 \times X_N) \text{ Mod } (30269). \tag{C.2}$$

$$Y_{N+1} = (172 \times Y_N) \text{ Mod } (30307). \tag{C.3}$$

$$Z_{N+1} = (170 \times Z_N) \text{ Mod } (30323). \tag{C.4}$$

$$RV = \left(\frac{X_{N+1}}{30269} + \frac{Y_{N+1}}{30307} + \frac{Z_{N+1}}{30323}\right) \text{ Mod } (1.0). \tag{C.5}$$

The Wichmann-Hill algorithm is based on three independent number generators (Equations C.2, C.3, and C.4). These generators have an individual cycle length of roughly 30000 numbers each. However, when the result from the three generators is combined in Equation C.5, a random variable is generated that is much longer, roughly 7×10^{12} elements. Because the algorithm is generated using relatively short sequences, the overall long periodicity is achievable with a number generator

that needs to handle integers of values ranging up to 52125632, or integer arithmetic of up to twenty-three bits. This range value is well below the typical thiry-two bit integer arithmetic available in today's desktop computers.

The resulting value from the Wichmann-Hill algorithm is a uniformly-distributed random variable with values between zero+ and 1. This algorithm produces a full sequence; therefore all values for any of the three variables–except zero–are valid starting points. The highest values that the algorithm can handle for the seeds are 30269, 30307, and 30323 for X, Y, and Z, respectively.

The Wichmann-Hill algorithm is an excellent number generator (tests are presented in the validation section of the paper). This generator will be used in the simulation as the basis for all uniform random number generators needed in the simulation.

C.2 Pseudo-Random Binary Sequences

Data that must be conveyed over a communications channel should be completely random. If there is any correlation at all within the data, it is not pure data but data with redundancies. In an effort to randomize a sequence that is pure data, it is necessary to use a random binary sequence. A pseudo-random sequence that emulates the statistical characteristics of the intended random sequence can be created for this purpose. For these cases, a linear feedback shift register (LFSR) of the form seen in Figure C.1 can be used as the pn-generator.

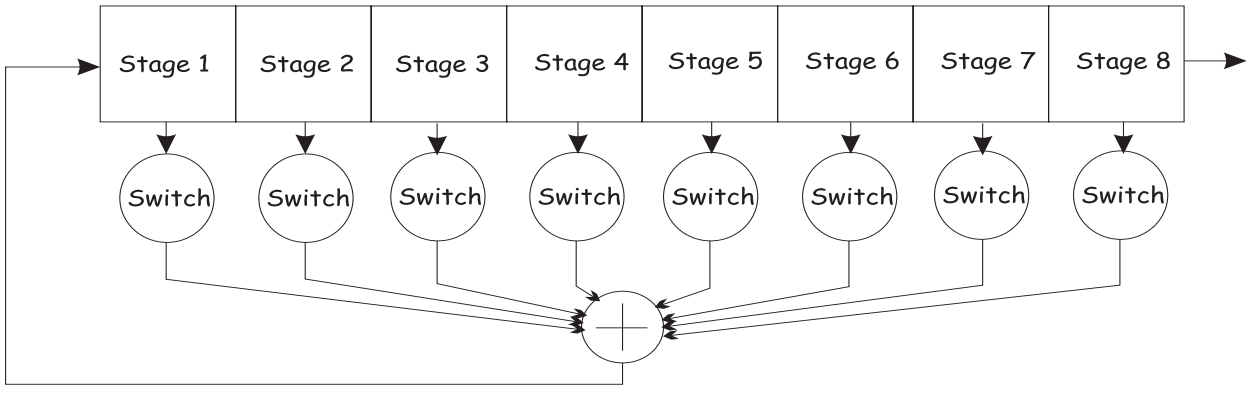

Figure C.1: Eight-Stage Linear Feedback Shift Register

The switch in each connection between a delay buffer and the feedback XOR determines whether the line is open or closed. Table C.1 shows polynomials for maximum length PN sequences [18]. Assuming that the polynomial selected produces a maximum-length sequence, the sequence length is $2^N - 1$ where N is the number of shift registers in the buffer.

Table C.1: Linear Feedback Shift Register Polynomials

Length	Sample Polynomial
2	[1,2]
3	[1,3]
4	[1,4]
5	[2,5]
6	[1,6]
7	[3,7]
8	[2,3,4,8]
9	[4,9]
10	[3,10]

Table C.1 is an incomplete list of available polynomials, providing a sample of polynomials available in the lower orders. For a more complete list of PN generator polynomials, see page 285 in [18].

Correlation is calculated by comparing the sequence with a phase-shifted copy of itself. The sequence reaches maximum correlation when the copy's phase shift is zero, and the sequence is perfectly correlated with itself. The spectrum of the correlation is very wide and equal at all frequencies, i.e., it is white. Validation of the pn-sequence generated in the simulation is shown in the simulation validation chapter.

C.3 Random Variable Transformation

The generation of a uniformly-distributed random variable is a straight-forward process. This random variable, although useful for several applications, needs to be modified to simulate the channels that the simulation must recreate. The channel being modeled is affected by Additive

White Gaussian Noise (AWGN) and may be subjected to Rician and Lognormal fading. There are also other channels that suffer from Rayleigh fading. Therefore, it is important to create four types of random variable distributions: Gaussian, Lognormal, Rayleigh, and Rician. The following sections will describe in detail the necessary algorithms to generate these random variables.

C.3.1 Gaussian

Gaussian random variables can be easily derived from uniform random variables. The Box-Mueller algorithm [18] can convert two independent uniform random variables into two independent Gaussian random variables.

$$X = \cos(2\pi U_2)\sqrt{-2\ln(U_1)}. \tag{C.6}$$

$$Y = \sin(2\pi U_2)\sqrt{-2\ln(U_1)}. \tag{C.7}$$

Both Equations C.6 and C.7 generate independent random variables from the same set of uniform random variables; U_1 in C.6 is the same as U_1 in C.7.

The Gaussian random variable is used as the building block for the next set of random variables.

C.3.2 Lognormal

A Lognormal random variable is one in which the distribution seen in a log-scale looks Gaussian.

$$X_{Lognormal} = 10^{\frac{\cos(2\pi U_2)\sqrt{-2\ln(U_1)}}{10}}. \tag{C.8}$$

$$Y_{Lognormal} = 10^{\frac{\sin(2\pi U_2)\sqrt{-2\ln(U_1)}}{10}}. \tag{C.9}$$

Equations C.8 and C.9 show the method to generate two independent Lognormal random variables based on two independent, uniform random variables. The process is the same as in Equations C.6

and C.7 with the extra step of taking the inverse-log of the resulting random variable.

C.3.3 Rayleigh

A Rayleigh distributed envelope can be generated based on Clarke's model [6], which makes several assumptions on the character of the multipath elements arriving at the terminal. Although these assumptions are invalid in special cases, such as when using highly directive antennas, the model applies to the general case.

One way of looking at a Rayleigh distribution is by drawing two orthogonal independent Gaussian random variables on a two-dimensional axis. A circle is drawn on this graph, centered at the origin. Every sample is defined by the combination of two Gaussian points on two axis. The Rayleigh random variable is the distance between the origin and the point defined by the two Gaussian random variables.

C.3.4 Rician

A Rician distribution is a special case of the Rayleigh distribution. The only difference between the Rician and Rayleigh channel models is a strong line-of-sight (LOS) component. In the Rician distribution, there is a strong LOS component that is occasionally attenuated by multipath elements. This LOS component is represented in the distribution by changing the mean of the variable.

Bibliography

[1] S. Y. Seidel and H. W. Arnold, "Propagation Measurements at 28 GHz to Investigate the Performance of Local Multipoint Distribution Service (LMDS)," in *IEEE Global Telecommunications Conference*, 1995.

[2] Y. Wang and Q.-F. Zhu, "Error Control and Concealment for Video Communication: A Review," *Proceedings of the IEEE*, vol. 86, May 1998.

[3] R. Peregrino, *Technical Specification 1.3, Part 8.* Digital Audio-Visual Council, 1997. www.davic.org.

[4] ISO/IEC-13818/2, *Generic Coding of Moving Pictures and Associated Audio; Part 2: Video.* Technical Report, International Standardization Organization/International Electrotechnical Commission, Geneva, Switzerland, 1995.

[5] P. B. Papazian, M. Roadifer, and G. A. Hufford, "Initial Study of the Local Multipoint Distribution System Radio Channel," NTIA Report 94-315, US Department of Commerce, 1994.

[6] T. S. Rappaport, *Wireless Communications, Principles & Practice.* Prentice Hall PTR, 1996.

[7] P. B. Papazian and G. A. Hufford, "Time Variability and Depolarization of the Local Multipoint Distribution Service Radio Channel," in *1997 Wireless Communications Conference*, 1997.

[8] W. Luo and M. E. Zarki, "Analysis of Error Concealment Schemes for MPEG-2 Video Transmission over ATM Based Networks," in *Proceedings of SPIE - The International Society for Optical Engineers*, vol. 2501, pp. 1358–1368, 1995.

[9] N. Lodge and D. Wood, "New tools for evaluating the quality of digital television - results of the mosaic project," in *Proceedings of the International Broadcasting Convention*, pp. 323–330, 1996.

[10] D. K. Fibush, "Video Testing in Modern Television Systems," in *Proceedings of the International Broadcasting Convention*, pp. 316–322, 1996.

[11] P. G. J. Barten, "Evaluation of CRT displays with the SQRI method," in *Proceedings of the SID*, vol. 30, pp. 9–14, 1989.

[12] P. Mouroulis and H. Zhang, "Visual instrument image quality metrics and the effects of coma and astigmatism," vol. 9, pp. 34–42, January 1992.

[13] I. E. G. Richardson and M. J. Riley, "Intelligent Packetising of MPEG Video Data," in *Proceedings of SPIE - The International Society for Optical Engineers*, vol. 2501, pp. 1388–1395, 1995.

[14] R. Matzner, P. Eck, and X. Changsong, "On MPEG-2 Decoding of Noisy Input Data," in *Proceedings of 3rd IEEE International Conference on Image Processing*, pp. 751–754, 1996.

[15] T.-C. Chen, "A real-time software based end-to-end wireless visual communications simulation platform," in *Proceedings of SPIE - The International Society for Optical Engineers*, vol. 2501, pp. 1068–1074, 1995.

[16] W. Luo and M. E. Zarki, "MPEG2Tool: A Toolkit for the Study of MPEG-2 Video Transport Over ATM based Networks," in *Proceedings of SPIE - The International Society for Optical Engineers*, vol. 2668, pp. 356–364, 1996.

[17] K. Shuaib, H. Elsayed, T. Saadawi, M. Lee, S. Gringeri, B. Khasnabish, and B. Basch, "Experimental and Simulation Results in the Transport of MPEG-2 Video over ATM," in *Proceedings of SPIE - The International Society for Optical Engineers*, vol. 3231, pp. 168–176, 1997.

[18] M. C. Jeruchim, P. Balaban, and K. S. Shanmugan, *Simulation of Communication Systems*. Plenum Press, 1992.

[19] J. G. Proakis, *Digital Communications*. McGraw-Hill, 3 ed., 1995.

[20] H. Stark and J. W. Woods, *Probability, Random Processes, and Estimation Theory for Engineers*. Prentice Hall, 2 ed., 1994.

[21] W. J. Ebel and W. H. Tranter, "The Performance of Reed-Solomon Codes on a Bursty-Noise Channel," *IEEE Transactions on Communications*, vol. 43, no. 2, 1995.

[22] D. R. Pauluzzi and N. C. Beaulieu, "A Comparison of SNR Estimation Techniques in the AWGN Channel," in *IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing Proceedings*, 1995.

[23] S. Lin and D. J. Costello Jr., *Error Control Coding: Fundamentals and Applications*. Prentice Hall, 1983.

[24] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*. Prentice Hall, 1995.

[25] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Prentice Hall, 1 ed., 1989.

Vita

Max Robert was born on April 24, 1973, in Buenos Aires, Argentina. In 1991, he received the Albert W. Smith Scholarship at Case Western Reserve University in Cleveland, Ohio, where he received his Bachelor of Science Degree in Electrical Engineering and Applied Physics in May of 1996. During his undergraduate program, he did three internships at Westinghouse Electric in the summers of 1991, 92, and 93 where he worked on several projects, from system automation to DSP algorithm implementation. During 1994, he worked for eight months at Sensis in DeWitt, New York, where he worked on a radar system upgrade for the United States Marine Corps. In 1995 and 1996, he worked for Keithley Instruments in Cleveland, Ohio, where he participated on the development of a prototype semiconductor testing device. In the fall of 1996, he joined the Master of Science Program at Virginia Tech. During the fall and spring of the academic year 1996 and 1997, respectively, he was a Graduate Teaching Assistant for the Bradley Department of Electrial and Computer Engineering. He has been working at the Mobile and Portable Radio Research Group as a Graduate Research Assistant since May of 1997. His research interests include system design, error correction techniques, video coding, system modeling and simulation, process automation, and general programming.