Studies in Error Correction Coding

Hannah Kirse

A Senior Thesis submitted in partial fulfillment
of the requirements for graduation
in the Honors Program
Liberty University
Spring 2014

Acceptance of Senior Honors Thesis

This Senior Honors Thesis is accepted in partial fulfillment of the requirements for graduation from the Honors Program of Liberty University.

_____

Terry Metzgar, Ph.D.
Thesis Chair

_____

Kyung Bae, Ph.D.
Committee Member

_____

Monty Kester, Ed.D.
Committee Member

_____

Brenda Ayres, Ph.D.
Honors Director

_____

Date

Abstract

For a proper understanding of the implementation of error correction coding schemes, a basic knowledge of communication channels and networks is necessary. Communication channels incur several types of errors, including noise and signal attenuation. Consequently, the benefits of a particular error control scheme are determined by the errors which occur most frequently. First, the types of transmissions across which errors occur will be considered. Subsequently, the types of errors that can appear during these transmissions and a short discussion of the cause of errors are necessary to understand the several types of errors that can occur. Afterward, the implementation of several major coding schemes will be discussed, including block codes, linear codes, and convolutional codes. Convolutional codes will specifically be discussed in terms of turbo codes and low-density parity check codes. Lastly, research of error correction coding schemes will involve several kinds of resources, including textbooks, journal articles, and technical publications. These resources will be used for the understanding of a practical implementation of an error correction coding scheme.

Studies in Error Correction Coding

Introduction

Communication and transportation networks provide several services to users. These types of services can range from radio and telephone to television and streaming video. Through these services, users are allowed to gather information in large volumes at particularly high speeds. Typically, this information is passed through communication channels over long distances. In an ideal situation, only data is transmitted across the channel. However, most communication channels experience a certain level of interference and noise. There are two kinds of data transmissions across channels: parallel and serial.

Data Transmissions

Parallel data transmissions. During a parallel transmission, all bits of data are transmitted simultaneously on separate communication lines. For this communication method, there is a direct correlation between the number of transmitted bits and the number of lines used. With each clock pulse, each group of bits is transmitted across the data lines. While initially a parallel data transmission appears to be more efficient than serial, this is not the case. In practice, the parallel data transmission operates based off of N communication channels. In contrast, a serial transmission only needs one communication channel, therefore reducing the over cost of a serial transmission. Consequently, serial data transmissions are used more frequently (Forouzan, 2001).

Serial data transmissions. When transmitting data between physically separate devices, especially if the devices are a long distance apart, it is more efficient to use a single pair of transmission lines. Serial transmissions allow various bits of data to be

transmitted one after another across one communication line. Each bit in a sequence is sent with each clock pulse. Serial transmissions are characteristically used for communication over long distances and computer networks. Within a computer, the hardware often transmits data in parallel. Consequently, conversion devices must be implemented to transfer the parallel data to serial data. The sender implements a parallel to serial conversion device and the receiver utilizes a serial to parallel conversion device. There are two types of serial transmissions, synchronous and asynchronous. Both of these types of serial transmission use bit synchronization. Bit synchronization is required to determine the start and end of a data transmission. This method also provides control of timing for the sending and receiving devices (Forouzan, 2001).

Asynchronous transmissions. An asynchronous transmission sends one character at a time. During this type of transmission, each character is either a literal character or a control character. It is referred to as asynchronous because the sender and receiver of the transmission do not need to be in sync. However, the receiver must be synchronized with the incoming stream of bits. Bit synchronization is implemented using a start and stop bit. With the addition of the start and stop bits, more bandwidth is also consumed in this type of transmission. This type of transmission is best suited for internet traffic and keyboard devices (Stallings, 2006).

Synchronous transmissions. In comparison to asynchronous transmission, synchronous transmission does not use start and stop bits. Instead, there are no gaps between the various bytes in the data stream. In this type of serial transmission, the bit stream is combined into longer frames which contain multiple bytes. Bit synchronization uses timing to receive and send each bit. Since the bits are sent to the receiver without

any gap or start and stop bits to distinguish, it is the responsibility of the receiver to reconstruct the information as it was originally sent. The receiver and sender must also operate at the same clock frequency in order for the data to be received error free (Stallings, 2006).

Error Types

Single-Bit Errors

Errors in a transmission can be categorized in two main categories, single-bit errors and burst errors. A single-bit error is an isolated event that alters one bit but not any surrounding bits. Single bit errors can happen in parallel transmissions where all the data bits are transmitted using separate wires. They are less likely to occur in serial transmission. While single-bit errors can negatively affect a message transmission, the probability of such an error occurring is relatively low. Consequently, burst errors are often considered to have a greater affect and are the focus of many error detection and correction schemes (Forouzan, 2001).

Burst Errors

Errors within communication channels do not appear evenly distributed across time. They appear in bursts and, as a result, are referred to as burst errors. A burst error is a group of bits in which two successive erroneous bits are always separated by less than a given number of correct bits. The length of the error is measured from the first changed bit to the last changed bit. While single-bit errors usually occur in the presence of white noise, burst errors are caused by impulse noise and fading. Burst errors are also more likely to occur in a serial transmission. This is because the noise occurs for longer durations and, therefore, affects more bits (Forouzan, 2001).

Sources of Errors

In electronic communications, noise is described as a random fluctuation in an electrical signal. Since noise signals are random, they cannot be directly predicted. Instead, statistical models are used to predict the average and variance for a signal. Within a communication system, noise is a significant source of errors and produces disturbances of a signal in a communication channel. Unfortunately, noise is unavoidable at any non-zero temperature. Since noise is both random and unavoidable, it was necessary to create a way to limit or prevent the corruption of data through a communication channel. Error control schemes have been created to help reduce the unpredictable effect of noise on a signal (Lathi & Ding, 2009).

Noise

Thermal noise. Noise can be divided into four separate categories: thermal noise, intermodulation noise, crosstalk, and impulse noise. However, this paper will focus on thermal noise, crosstalk, and burst noise or impulse noise. Thermal noise, or white noise, is the result of the agitation of electrons by heat. This stimulation of electrons results in unwanted currents or voltages within an electronic component. Often times, thermal noise is referred to as white noise because it is uniformly distributed across the bandwidths used in communications systems. The presence of thermal noise inhibits the receiving equipment's ability to distinguish between incoming signals. Because thermal noise cannot be ignored or eliminated, an upper bound is placed on the performance of a communication system.

Crosstalk. Crosstalk is a type of noise that occurs when an electrical signal is picked up by an adjacent wire. It is the result of an unwanted electrical coupling between

signals paths. Typically, crosstalk is recognizable as pieces of speech or signal tones leaking into a person's telephone connection during a conversation. One type of crosstalk, called near-end crosstalk, occurs when a signal on the transmit pair is so strong that it radiates to the receive pair. A direct consequence of this radiation is that the receiver cannot correctly interpret the real signal. Like thermal noise, crosstalk has a reasonably predictable and constant magnitude. Therefore, it is possible to construct a communication system which has the ability to cope with the noise. However, noise, such as burst noise, is more difficult to handle because it is unpredictable and appears erratically.

Burst noise. Burst noise, or impulse noise, is another significant source of signal corruption within a communication channel. Burst noise is the result of sudden step-like transitions between at least two discrete voltage or current levels. Impulse noise, unlike thermal noise, is discontinuous. This type of noise results from irregular pulses or noise spikes of short duration and relatively high amplitude. While impulse noise is not as impactful on analog data, such as voice transmissions, it is a primary source of error for digital data communication. Since burst noise is often the result of signals induced by external sources, such as lightning, switching equipment, or heavy electrically operated machinery, it quite frequently interrupts surrounding signals in communication channels (Lathi & Ding, 2009).

Gaussian Noise

Gaussian noise is statistical noise which has a probability density function equal to that of the normal distribution. This means that the probability of a signal falling within a particular range of amplitudes is considered to be normally distributed. Channels

experience a constant interference from other signals, background noises, and a variety of other sources. Gaussian noise is the most efficient for simulating background noise, amplifier noise from transceivers, or signals from other communication systems within the same frequency band. As a result, Gaussian noise is combined with a channel for simulating real-life circumstances (Lathi & Ding, 2009).

An additive white Gaussian noise channel, or AWGN, is a model used to provide a simple mathematical model of a communication system. This model is helpful to understand how a communication system works before introducing other occurrences such as fading, frequency selectivity, interference, nonlinearity, or dispersion. While natural additives in channels may be due to several different causes, the central limit theorem says the cumulative effect of a large number of small random effects on the channel will be approximately normal. In other words, in mathematical models, it is easier to assume that all noise is Gaussian noise. White noise is a signal which has a constant power spectral density, which means any signal within a frequency band contains equal power. White Gaussian noise is a combination of the properties of white noise and Gaussian noise. It is considered to be a good approximation and the most realistic for modeling real-world situations. The AWGN model was created as the result of a linear combination of white Gaussian noise with wireless channel. It provides a performance upper-bound for general distortive channels.

Without noise, data communications would be relatively simple. In an ideal communication with no noise, to increase the rate at which data is transferred the bandwidth, or range of frequencies the energy of a signal is contained in, must be increased. Unfortunately, this is not possible because increasing the rate at which data is

sent would cause more errors. This is because when the data rate is increased, more bits are affected by a given pattern of noise. With the faster bit rate, more bits occur during a noise spike, which causes more errors within a bit string. Claude Shannon developed a formula which helps engineers to discover how efficient their communication channels can be.

Noisy-Channel Coding Theorem

The channel coding field was started around 1948 with the publishing of Claude Shannon's paper on his noise channel coding theorem. The goal of channel coding was to create practical coding schemes to approach channel capacity. Practically speaking, the channel capacity or "Shannon limit" has been reached through the implementation of turbo codes and low-density parity-check codes. Before Shannon's theorem was published, it was the common belief that in order to obtain relatively error-free communication, the rate at which the data was transmitted had to approach zero. Shannon discovered that this was not true, and the boundary between reliable and unreliable communication was a line crossing through the R-axis in the (R, p_b) space, with p_b the bit-error probability, at a non-zero value. Therefore, for any channel, a code exists that makes it possible to communicate with an arbitrarily small probability of error. The boundary Shannon described demonstrates where reliable communication at any rate is impossible and where reliable communication at any rate is possible. Shannon's noisy-channel coding theorem states, "Information can be communicated over a noisy channel at a non-zero rate with arbitrarily small error probability." Summarized, this theorem says for any given degree of noise contamination of a communication channel, it is possible to

transfer discrete data nearly error-free up to a calculable maximum rate (Lathi & Ding, 2009).

Shannon Limit

In addition to the noisy-channel coding theorem, Claude Shannon established the Shannon limit. The Shannon limit represents the theoretical maximum capacity of an ideal bandwidth. He established how the relationship of bandwidth and the ratio of signal to noise power affected the channel capacity. The Shannon limit is represented as $C = B_w * \log_2(1 + \frac{S}{N})$, where C = max channel capacity, B_w = channel bandwidth, S = signal power, N = noise power (Shannon limit., 2001).

The Shannon limit is not practically achieved for maximum bandwidths; in fact, only much lower bandwidths are more attained. This is due to the formula not accounting for impulse noise or several kinds of distortion. Also, coding length and complexity do not allow current technology to achieve the Shannon limit even in an ideal white noise environment. However, this limit is still frequently used as a metric for the performance of practical communication methods.

Error Detection

Anything from the proper operation of the internet to talking to another person on the phone would not be possible without error detection and correction codes. The goal of any digital communication is to transmit and receive information without data loss or the corruption of data. The fundamental idea behind error correction coding is to add redundancy to the transmitted signal. The added redundancy will allow the receiver to detect and correct errors. There are three kinds of redundancy checks: vertical,

longitudinal, and cyclic. These checks are implemented by codes to detect as many errors as possible within a transmission.

Vertical Redundancy Check

Often referred to as a parity check, the vertical redundancy check is the most commonly used and least expensive of the three checks. This technique involves the addition of a redundant bit, or parity bit, to every data unit. By adding parity bits, this ensures the total number of ones in a data transmission is even. While some systems use an odd-parity check, the majority of the systems implement an even-parity check. Through this technique, a vertical redundancy check can detect single-bit errors as well as detect an odd number of burst errors. Unfortunately, this check cannot detect errors which result in an even number of bits changed. Since a vertical redundancy check only checks for whether or not the number of ones in a transmission is even, an error which results in an even number of ones in the data will be considered error-free (Forouzan, 2001).

Longitudinal Redundancy Check

Instead of sending bits in one row, the longitudinal redundancy check organizes bits into a table. For each column, a parity bit is created. These new bits are used to form the last row of the previously assembled table. Thus, the last row of the table is referred to as the parity row. A longitudinal redundancy check increases the probability of the detection of a burst error. Regrettably, there is one circumstance in which the longitudinal redundancy check fails and is unable to detect an error. If the same number of bits is damaged in exactly the same position in another row, this check will fail to register an error has occurred (Hughes, 1996).

Cyclic Redundancy Check

While vertical redundancy checks and longitudinal redundancy checks are useful, they are not efficient enough for the detection of errors. The cyclic redundancy check offers accuracy through the use of polynomial division. For this reason, the code that implements a cyclic redundancy check is often referred to as a polynomial code (Widjaja & Leon-Gracia, 2000). In this technique, each bit in a message is considered as a coefficient of a polynomial. Thus, for each message, a message polynomial is created. Combined with the check bits, the message polynomial creates a unique cyclic codeword. When the message has been received, the received bits are divided by the generator polynomial and the remainder is checked to see if its value is 0. The generator polynomial is the polynomial that the sender and receiver have agreed upon to represent their transmissions (Warren, 2012). If the remainder is 0, then the message has been successfully transmitted. A cyclic redundancy check will be able to detect burst errors of up to r number of errors, where r is the number of bits in the generator polynomial (Forouzan, 2001).

Error Correction

While error detection is helpful in data transmissions, error correction is essential to a successful transmission. Error correction allows a distorted transmission to be returned to its original transmitted form. With the introduction of redundancy, a receiver can detect some errors. Additionally, these errors can be corrected to return a transmitted message to its original form. This paper will discuss the two types of error correction coding: automatic repeat request (ARQ) and forward error correction (FEC). ARQ attempts to correct errors via retransmission, whereas FEC attempts to correct

transmissions at the receiving end of a transmission without the need to retransmit a message (Gallo & Hancock, 2002).

Automatic Repeat Request

During the process called error control coding, redundant information is added to a message that is to be transmitted. After the message is received, the redundant information is to be used to detect and potentially correct errors that occurred in the message during transmission. Error control coding has two classifications: automatic repeat request and forward error correction (Garg, 2007). ARQ is an error-detection system which checks a message for errors. If an error is found, the receiver will inform the sender, and the section of the message that contains the error will be resent (Automatic repeat request (ARQ), 1999). The receiver in an ARQ system performs error detection on all received messages. Consequently, all responsibility for detecting errors lies with the receiver. The ARQ must be simple enough for the receiver to handle it, yet powerful and efficient enough so the receiver does not send erroneous messages to the user. Based on retransmission, there are three basic ARQ schemes: stop-and-wait, go-back-N, and selective repeat.

Stop-and-wait. In the stop-and-wait scheme, the transmitter sends a codeword to the receiver and waits for acknowledgement. A positive acknowledgement means the transmitted codeword has been successfully received, and the transmitter sends the next codeword in queue. When a receiver sends a negative acknowledgement signal, this implies an error has been detected in a codeword. Once the transmitter receives this signal, it attempts to resend the invalid codeword. These retransmissions will continue until the receiver sends a positive acknowledgement signal to the sender (Lin, Costello, &

Miller, 1984). While this scheme is very simple, data is transmitted in only one direction. This type of one-way transmission does not meet the qualifications for the high-speed modern communication systems of today (Gibson, 2002).

Go-back-N. In the go-back-N scheme, codewords are continuously transmitted by the transmitter in order. N is size of the "window" from which the transmitter can send codewords. It is important for N to be as large as possible within the bounds of the receiver's ability to process packets and the number of codewords used to verify a transmission. The acknowledgement signal for a codeword typically arrives after a round-trip delay. A round-trip delay is defined as the time between the transmission of a codeword and the reception of an acknowledgement signal for a codeword. The transmitter continues to send N-1 codewords during this interval. When the sender receives a negative acknowledgement signal designating an erroneous codeword, it goes back to that codeword and resends that codeword and all N-1 following codewords. A significant disadvantage of this method is when a receiver detects an error all following codewords are discarded (Lin, Costello, & Miller, 1984).

Selective repeat. The selective repeat method was developed to overcome the disadvantages of both the stop-and-wait and the go-back-N schemes. Similarly, codewords are sent continuously; however, if a codeword is negatively acknowledged, only that codeword is retransmitted. In this system, a buffer is needed to store the error-free codewords that follow the incorrect codeword. A buffer is a region in memory that is used to temporarily keep data from continuing while it is in the process of transmission. The buffer is necessary because data must generally be received in order. A size for the buffer must be chosen to be large enough so data overflow does not occur and codewords

are not lost (Lin, Costello, & Miller, 1984). Complications with this technique can arise when a second codeword is found invalid while the first corrected codeword has not yet been received by the buffer. Proper buffering must be maintained by the transmitter which must have the necessary logic to handle several out of sequence codewords (Gibson, 2002).

Forward Error Correction

In addition to error-detection, FEC is a method that attempts to correct errors found. Extra bits are added, according to set algorithm, to the transmission of a message. These extra bits are received and used for error detection and correction, eliminating the need to ask for a retransmission of data (Forward Error Correction (FEC), 1999). One of the first codes to appear was created by Richard Hamming. Hamming developed an infinite class of single-error-correcting binary linear codes. This code was said to form an exhaustive partition of binary n-space. Shortly after, Marcel Golay published a comprehensive binary linear (23, 12, 7) triple-error-correcting code as well as a similar (11, 6, 5) double-error-correcting ternary code. These three codes are known to be the only nontrivial "perfect" linear codes in existence today. Essentially, there are two kinds of error correction codes implemented in FEC, linear codes and block codes.

Linear codes. A linear code, or binary linear code, implements linear combinations of codewords as a means of encoding a transmitted bit stream. This code takes a group of K message bits and produces a binary codeword that consists of N bits. The extra N-K parity-check bits, provided for redundancy, are determined by a set of N-K linear equations. Each parity check bit can be represented in an equation such as the following: $b_{k+1} = a_{11}b_1 + a_{12}b_2 + \cdots + a_{1k}b_k$ (Widjaja & Leon-Gracia, 2000).

Block codes. Block codes organize a stream of bits in binary format, referred to as the message bits, into blocks of length K. Since each block of bits is of length K, there is a set of 2K possible messages that can be transmitted. The bit string encoder converts the set of 2K possible blocks of K bits into another set of 2N longer blocks of N bits. From this larger set of 2N possible codewords, N codewords are chosen to be sent in the transferred stream. The extra (N-K) bits, or parity check bits, are the redundancy addition to the bit stream before it is communicated. The final result is a codeword which is transmitted, corrupted, and decoded separately from all other codewords.

A block code has a code rate of K/N, which refers to the level of redundancy that is applied to a block code. A code rate of K/N represents the ratio of the number of message bits to total number of coded bits. An (N,K) block code for a channel Q is a list of $S = 2^K$ codewords each of length N. The number of codewords S is an integer, however, the number of bits specified by choosing a codeword, $K = \log_2 S$ is not necessarily an integer.

When designing a block code to be used, there are several key considerations. The values of N and K must be chosen so that the minimum distance of a codeword is as large as possible. The distance of a codeword is considered the number of bits that are different in a codeword after transmission. Therefore, the minimum distance of a codeword signifies the number of single-bit errors that can be detected by an error code. Additionally, the number of extra bits should be as small as possible to reduce the bandwidth of a transmission. Finally, the number of extra bits must be a large as possible to reduce the error rate. Consequently, a critical relationship exists between the bandwidth and error rate of a transmission (Stallings, 2006).

Linear block codes. A linear code is an error-correcting code in which any linear combination of codewords is also a codeword. There are several block codes which also belong to the class of linear codes, such as Hamming codes, Reed-Solomon codes, Hadamard codes, Expander codes, Golay codes, and Reed-Muller codes. This set of codes is referred to as linear block codes because they belong to both classes of codes.

In linear block coding, a linear encoding and decoding scheme translates a sequence of source bits into a transmitted sequence of bits. This scheme inserts extra bits, called parity-check bits, to add redundancy and improve reliability. A sequence of K information symbols is encoded in a block of N symbols, where N > K, and then transmitted over the channel. First, K information bits enter an encoder, and the encoder generates a sequence of coded symbols of length N to be transmitted over the channel. For this codeword, or transmitted sequence, N must be greater than K to guarantee uniqueness between each codeword and each of the possible 2^K messages (Garg, 2007). The most well-known example of a linear block code is the (7,4) Hamming code. For every 4 source bits transmitted, the Hamming code transmits an additional 3 parity-check bits. This redundancy guarantees at most one error can be corrected (MacKay, 2003).

Cyclic codes. Cyclic codes are error-correcting codes that became popular around the time of block codes because of their use of finite-field algebraic structures. Cyclic codes are considered to be a subclass of linear block codes due to the additional property of invariance under a cyclic shift of n-tuple codewords. Shifts are also called "end-around" movements (Costello & Forney, 2007). In other words, if a codeword such $c = (c_1, c_2, \dots, c_{n-1}, c_n)$ exists, then $(c_2, c_3, \dots, c_n, c_1)$ as is $(c_3, c_4, \dots, c_1, c_2)$ and so on are also considered codewords. Therefore, each (n,k) codeword can be represented by a code

polynomial: $c(x) = c_1x^{n-1} + c_2x^{n-2} + \cdots + c_n$. A code generator polynomial of degree n-k divides $x^n - 1$, which is found in the set of all codewords. Additionally, each code polynomial can be represented by the product of the code generator polynomial, g(x), and the data polymonial, d(x), where $c(x) = d(x)g(x) = (d_1x^{k-1} + d_2x^{k-2} + \cdots + d_k)g(x)$. This equation is also used to represent every codeword formed in a linear (n,k) code (Lathi & Ding, 2009). A Reed-Solomon code is a cyclic block code that can detect and correct multiple errors. First the data stream is broken into blocks which are further divided into a number of symbols whose size range from six to ten bits. The original data is comprised of a block of N-R symbols which is run through an RS encoder. The encoder adds R check symbols to form a codeword of length N (Garg, 2007). An (N,K) RS code can recover the original K source symbols if any K of the N transmitted symbols are received. By adding t check symbols to data, an RS code can detect any combination of t symbols with error or correct up to t/2 symbols. This is the reason why these codes excel in correcting erasure errors. Today, RS coding is a key component in compact discs, DVDs, and barcodes. If a two-dimensional barcode, such as a QR code, is damaged, RS code treats the damage as an erasure and attempts to read the correct, undamaged portion of the barcode and infer the rest of the barcode.

Convolutional codes. Similar to block codes, convolutional codes are also linear codes. However, data from the source stream is not divided into blocks; it reads and transmits data continuously. These transmitted bits are a linear function of the past source bits. Because the behavior of the code depends on previous data, convolutional codes are said to have memory. A linear-feedback shift-register of length k is first fed the present source bit. Afterward, every iteration through the shift register transmits one or more

linear functions of the state of the shift register. As a result, the final transmitted bit stream is the convolution of the source stream with a linear filter.

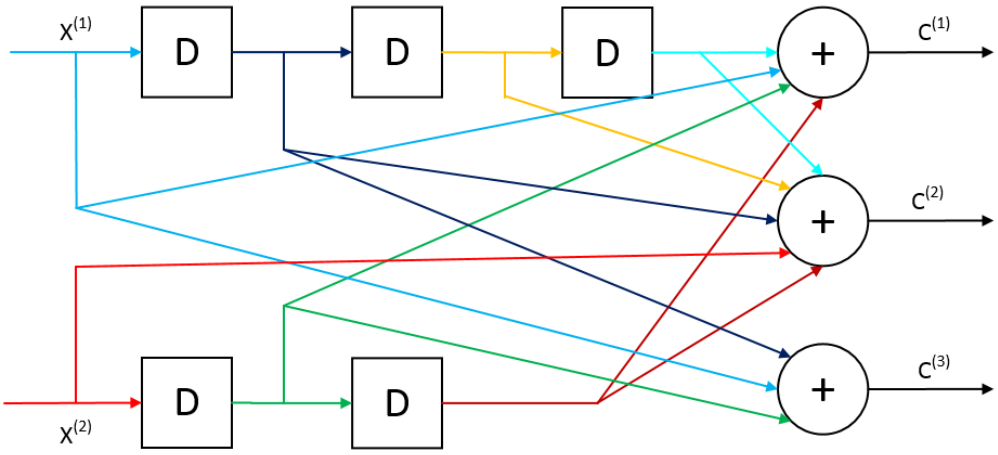

Figure 1. Data output for a convolutional code.

Figure 1 demonstrates a basic convolutional coding scheme. The original source bits of the data stream are represented as $x^{1)}$ and $x^{(2)}$. Each D represents a linear-feedback shift registers through which the data bits are passed. The final $c^{(n)}$ bits represent the final bit stream.

Convolutional code filters. Convolutional codes are broken down further into three sections based on the type of filter that is used. A systematic filter includes unmodified input in the output and nonsystematic does not. “Recursive” means one previously transmitted bit is fed back into the shift register along with the source bit. In a systematic nonrecursive system, one of the output bits will be identical to a source bit and the second transmitted bit is a linear function of the state of the filter. Nonsystematic nonrecursive filters are considered superior because their enhanced complexity gives them enhanced error-correcting abilities. A systematic recursive filter transfers one original input bit as an output bit and the second output bit is derived from a linear

function based on the state of a previous output bit. Since systematic recursive and nonsystematic nonrecursive codes define the same set of codewords, they are considered code-equivalent. However, there are significant differences in how their encoders behave. The nonrecursive encoder has a finite impulse response, meaning if a sting is inserted that contains all zeroes except a single one, the result will also contain a finite number of zeroes. Once the single inputted one has passed through all states of memory, an all-zero state will be returned to. In contrast, a recursive code has an infinite impulse response. The single bit containing a one will impact every future output because the output is dependent on both past and current inputs.

To decode a convolutional code, the Viterbi algorithm is the most efficient. The Viterbi decoding algorithm implements "maximum likelihood" decoding; it attempts to successfully identify the most plausible codeword. Classified as a non-iterative trellis-based decoder, by observing the received sequence the Viterbi algorithm finds the path through the tree with the largest value. This path is considered the best guess at what the original non-corrupted sequence was.

Concatenated codes. In 1966, Dave Forney introduced the idea of a concatenated code. This code would include a serial cascade of two linear block codes. The main idea of the concatenated code was that the inner and outer codes would be easy to encode and decode by themselves.

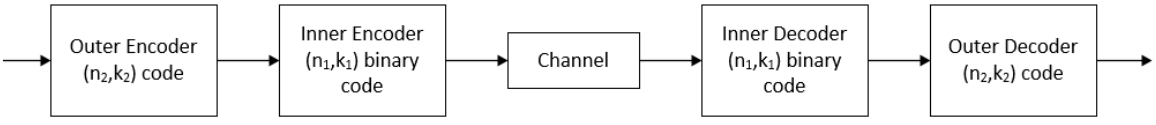

*Figure 2.*Encoding and decoding in a concatenated code.

As shown in figure 2, the original data stream passes first through a set of concatenated encoders. Next, the encoded stream is transmitted to a receiver over a channel. Finally, the encoded data stream is decoded at the receiver.

Through its design, concatenated codes have the ability to be longer and more powerful than a basic linear block code. The overall result was a long, powerful code with a simple decoder that could correct many random errors. An important result of Forney's idea was that by choosing the correct code, a concatenated coding scheme could operate near the Shannon limit with exponentially decreasing error probability (Costello & Forney, 2007). The most successful concatenation of two codes is the use of a Reed-Solomon outer code and a binary convolutional inner code (Lathi & Ding, 2009). This type of concatenated coding was the predecessor to what are now known as turbo codes.

Turbo codes. Turbo codes are considered a class of convolution codes because they are a concatenation of two convolutional codes. However, this new class of codes almost reaches the theoretical limits established by Shannon's theorem. A typical turbo code implements several parallel or serial concatenated RS code encoders separated by an interleaver. An interleaver gives a permutation of the information bit sequence and this new sequence is passed to the second RS code encoder as input. Interleaving performance gain is the term used to describe the relationship between increasing the size of the interleaver and decreasing frequency of errors (Albeitazzi, Ernst, Gappmair, Liva, & Papaharalabos, 2007).

An (N, K) turbo code is defined by several convolutional encoders and an equal number of interleavers. First, a string of K source bits is fed into each encoder. Next, the

order of the source bits are changed or altered in some way based on the encoder they were fed into. A turbo code encoder sends out three sub-blocks of bits.

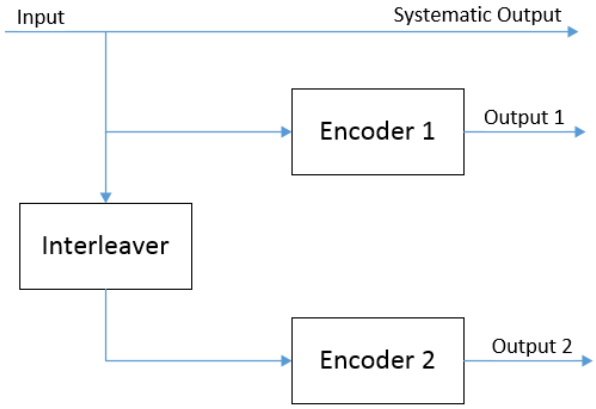

*Figure 3.*Turbo code encoder.

Figure 3 demonstrates how the source bits are passed through an interleaver and two separate encoders to generate three outputs. The first sub-block is the source data. The second and third sub-blocks are blocks of parity bits computed based on different convolutional coding schemes. The resulting transmitted codeword consists of K source bits, M_1 parity bits, and M_2 parity bits.

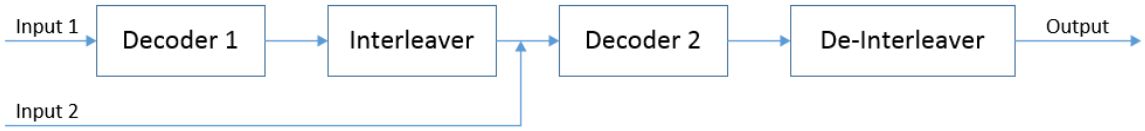

Figure 4. Turbo code decoder.

Finally, Figure 4 demonstrates how a turbo code is decoded when it has been received. The encoded bit stream is sent through a series of decoders and interleavers to output the original data stream to the receiver.

Turbo codes have exceptional performance with decoded error probabilities of around 10^{-5} (MacKay, 2003). The use of recursive convolutional encoder and interleavers is critical for turbo codes. This helps to make the turbo code appear to be more random

and as a result reduces the number of low-weight codewords. Low-weight codewords are are considered beneficial to turbo codes. Because of their attempt to maximize the minimum distance between two codewords, a turbo code can correct about half of the patterns of channel errors. (Costello & Forney, 2007). Turbo codes are also referred to as iterative codes because the decoder essentially works at making a series of guesses about what the encoded message is. By continually repeating this process, the error-rate will also decrease (Hardesty, 2010).

Soft decision and hard decision. A key component of a turbo code is the soft input, soft output decoder. Each decoder provides a soft output decision for the subsequent decoder. During hard decision decoding, the decoder interprets each bit as either definitely zero or definitely one. Typically, hard decision decoding compares a bit to an established threshold. If the bit is greater than the threshold, it is considered a one; otherwise, the bit is considered a zero. Comparatively, soft decision decoding measures the probability of an output being correct or reliable. Typically in soft-decision decoding, the decoder outputs a soft output decision which is a real number. This real number represents the probability a bit is still precise after being transmitted. For each binary value, the soft decision decoder also receives a confidence value. This confidence value is determined by a demodulator. The demodulator assigns a high confidence value if it believes it is a one and a low confidence value if it believes it is a zero. For the Viterbi decoder, the decoder receives soft information from a demodulator and outputs hard decision data. The decoder uses the soft information to determine if the given bit is a solid one or a solid zero and returns this hard decision. In turbo coding, a demodulated soft decision is fed into a soft input, soft output decoder. The corresponding output is fed

into the same decoder or another soft input, soft output decoder. Each iteration of this process changes the confidence value either higher or lower until a confident solution is achieved (Garg, 2007).

Low-density parity-check codes. Originally published in 1962 by Robert Gallager, the concept of low-density parity-check codes (LDPC) remained dormant for several decades. Gallager used a graphical representation of bit and parity-check sets of a regular LDPC code to describe the application of iterative decoding. With the publication of turbo codes, LDPC codes, the original iterative coding scheme, were rediscovered by McKay and Neal in 1996 (Moreira & Farrell, 2006). LDPC codes are designed as both a linear code and a binary block code. However, there are several key differences between block codes and LDPC codes. When a classical block code decodes a sequence, it attempts to make the decoding process as simple as possible. Although LDPC codes are considered to be similar to block codes, they are significantly different in how data is decoded. LDPC codes are decoded iteratively using a graphical representation of their parity matrix. As a result, an LDPC code is focused on the properties of its parity matrix. The parity matrix, by definition, is characterized as sparse. In other words, the parity matrix contains the smallest amount of zeroes as possible. The generator matrix for the code is found after the parity matrix is created. The generator matrix converts a message vector into a code vector, or codeword, via matrix multiplication.

To graphically represent an LDPC code, a Tanner graph is used. A Tanner graph consists of two sets of vertices; one set represents the codeword bits, or bit node, and the other set represents parity-check equations, or check nodes. An edge joins a bit node to a check node if the bit is included in the corresponding parity-check equation.

Consequently, the number of edges in a Tanner graph is equal to the number of ones in the parity-check matrix (Johnson).

Correction of Burst Errors

Interleaving is a technique frequently used for the correction of burst errors. The technique of interleaving is to disperse a large burst of errors over multiple codewords. As a result, each codeword is only responsible for the detection and correction of a fraction of the original erroneous segment of bits. Consequently, previously unusable random error correcting codes can be used to correct burst errors with precision.

There are two kinds of interleavers implemented in an interleaved code: block and random. Block interleavers input a block of bits and generate an output sequence in a systematic pattern. In contrast, random interleavers do not create an output which follows a set pattern. This is a generalized interleaver which can reorder the data bits inside the interleaver and output then in a fashion only known to the transmitter and receiver. Random interleavers are often more effective in repairing random and burst errors due to the smaller probability of receiving a burst of error bits in a codeword. Another factor of how well an interleaver can handle a burst of errors varies with the interleaver depth. The depth of an interleaver refers to the total memory length and directly corresponds to an interleaver's ability to handle longer bursts of errors. However, there is a cost based on the need for larger memory and delays due to longer encoding and decoding times (Lathi & Ding, 2009).

ARQ versus FEC

There are several key differences between ARQ and FEC coding. The functional purpose of the receiver varies for each coding scheme. ARQ coding allows only detection

of errors by the receiver. This results in slower data reception rates because the receiver must wait for the sender to retransmit the invalid data. ARQ is also favored if the size of the channel data transmitted across is unknown or varies. Conversely, FEC is unable to handle channels of unknown or varying size, because their encoders and decoders require prior knowledge of the channel capacity in order to be properly coded. Additionally, FEC allows both the detection and correction of errors by the receiver (Garg, 2007). Therefore, in transmissions such as loading a picture on the internet, ARQ techniques are preferred. Also, FEC is favored in transmission where real-time error correction is necessary, such as a telephone call or deep-space communications. On the other hand, if an ARQ method was implemented in a telephone conversation and an error was discovered, the retransmitted data would be sent too slowly to be helpful. FEC information is also frequently added to mass storage devices to enable recovery of data. For example, the second level of RAID implements the Hamming code as a data recovery method. In the end, both ARQ and FEC have their own advantages and disadvantages. However, the method chosen often relies on the type of transmission.

Applications of Error Control Codes

Error control codes have been applied to a variety of areas since its development. One of the earliest applications of error control codes was in the field of deep-space and satellite communication systems. Although these systems has nearly unlimited bandwidth, power was very expensive. Thus, it was critical for the systems to operate on as little power as possible. Consequently, early error control codes were focused on a small subset of the capacity curve. However, the deep-space channels were important because they were considered a perfect place to demonstrate the efficiency and potential

of codes. The memoryless AWGN channel, which forms the basis for the noisy channel coding theorem, was almost exactly recreated by the deep-space channels.

Data storage is the most well-known application of error control codes within the last fifty years. These codes are used in computer memory to increase the reliability. This application field presents different challenges for an error correction code than those encountered in deep-space communications. Encoding and decoding must be very quick so that the data read time is not drastically affected. The number of redundant bits must be minimized because the cost of memory is high. Additionally, different types of memory systems encounter various types of errors that are not seen in the average data transmission. Semiconductor memory systems often encounter byte errors as a result of a chip failure. Errors for audio signals which are usually found on CDs do not necessarily need to be corrected. Instead, these errors can be estimated using the surrounding data. Therefore, CD-ROMs have a greater error correction capability. Concatenated Reed-Solomon codes are able to correct burst errors of up to a length 4000 bits. Due to their efficiency, Reed-Solomon codes have been the standard for error correction on optical-disc memories such as CD-ROMS (Costello, Hagenauer, Imai, & Wicker, 1998). More recently, Reed-Solomon codes have been used for error correction on DVDs and Blu-Ray disc as well.

One of the most recent applications of error control coding is in mobile communications. Error control coding is necessary in mobile communications because mobile channels often have error rates of five to ten percent. Accordingly, decoders must work to be more efficient. Furthermore, bandwidth on mobile channels is expensive and

scarce. Thus, the redundancy bits that are typically added to a transmission must be limited.

Conclusion

The goal of any digital communication is to transmit and receive information without data loss or corruption. Consequently, anything from proper operation of the internet to talking to another person on the phone would not be possible without error detection and correction codes. For this reason, several error control techniques were introduced to reduce errors in data transmission where the errors are unacceptable. Redundancy is introduced into a message transmission through a vertical redundancy check, longitudinal redundancy check, or cyclic redundancy check. This redundancy increases the probability that an error will be found. After an error has been found, there are several techniques a receiver can implement to correct any errors within a transmission. ARQ techniques correct errors in a transmission by asking the transmitter to resend data. ARQ techniques can be broken into three categories: stop-and-wait, go-back-N, and selective repeat. The most efficient of the three categories is the selective repeat method.

FEC procedures correct errors based on parity bits or confidence values of a bit. FEC procedures are broken into two categories: block codes and convolutional codes. A frequently known example of a block code is the Hamming code. Turbo codes, while often mistaken as a separate category of codes, are actually a concatenation of usually two convolutional codes. Less than twenty years since the introduction of turbo codes and the rediscovery of LDPC codes, almost all digital communications involve coding which approaches Shannon’s theoretical limit.

References

Albeitazzi, G. C., Ernst, H., Gappmair, W., Liva, G., & Papaharalabos, S. (2007). *Digital signal communications*. New York City: Springer Publishing.

Automatic repeat request (ARQ). (1999). Retrieved from Focal Dictionary of Telecommunications: http://www.liberty.edu:2048/login?url=http://www.credoreference.com/entry/bhfidt/automatic_repeat_request_arq

Costello, D. J., Hagenauer, J., Imai, H., & Wicker, S. B. (1998, October). Applications of error control coding. *IEEE Transactions on Information Theory, 44*(6), 2531-2559.

Costello, D., & Forney, G. J. (2007, June). Channel coding: The road to channel capacity. *Proceedings of the IEEE, 95*(6), 1150-1177. Retrieved from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4282117&isnumber=4282113

Forouzan, B. (2001). *Local area networks*. New York: McGraw Hill.

Forward Error Correction (FEC). (1999). Retrieved from Focal Dictionary of Telecommunications: http://www.liberty.edu:2048/login?url=http://www.credoreference.com/entry/bhfidt/forward_error_correction_fec

Gallo, M., & Hancock, W. (2002). Computer communications and networking technologies. Pacific Grove, CA: Brooks/Cole Publishing Co.

Garg, V. (2007). Wireless communications and networking: an introduction. Burlington, MA. Retrieved from http://site.ebrary.com/lib/liberty/Doc?id=10188221&ppg=263.

Gibson, J. D. (2002). The communications handbook. Retrieved from http://radio-1.ee.dal.ca/~ilow/4540/readings/crc_ch14.pdf.

Hardesty, L. (2010, January 21). Explained: Gallager codes. MIT's News Office.

Hughes, L. (1996). Introduction to data communications: A practical approach. Jones & Bartlett Pub.

Johnson, S. J. (n.d.). Introducing low-density parity-check codes. Retrieved from http://materias.fi.uba.ar/6624/index_files/outline_archivos/SJohnsonLDPCintro.pdf

Lathi, B. P., & Ding, Z. (2009). Modern digital and analog communication systems (4th ed.). New York: Oxford University Press.

Lin, S., Costello, D., & Miller, M. (1984). Automatic-repeat-request error-control schemes. *Communications Magazine, 22*(12), 5-17.

MacKay, D. J. (2003). Information theory, inference and learning algorithms. Cambridge university press. Retrieved from http://www.cs.toronto.edu/~mackay/itprnn/book.pdf

Moreira, J. C., & Farrell, P. G. (2006). Essentials of error control coding. John Wiley & Sons, Ltd.

Shannon limit. (2001). Retrieved from Hargrave's Communications Dictionary.

Stallings, W. (2006). Data and computer communications. Upper Saddle River: Prentice Hall.

Warren, H. (2012). Hacker's delight (2nd ed.). Addison-Wesley Professional. Retrieved from http://www.hackersdelight.org/crc.pdf

Widjaja, I., & Leon-Gracia, A. (2000). Communication networks: Fundamental concepts and key architectures (1st ed.). Mcgraw-Hill College.