NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

DISSERTATION

ERROR CORRECTION CODE-BASED EMBEDDING IN ADAPTIVE RATE WIRELESS COMMUNICATION SYSTEMS

by

Peter M. B. Harley

September 2019

Dissertation Supervisors: John C. McEachen
Murali Tummala

Dissertation Supervisors:	John C. McEachen Murali Tummala

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2019	**3. REPORT TYPE AND DATES COVERED** Dissertation

4. TITLE AND SUBTITLE ERROR CORRECTION CODE-BASED EMBEDDING IN ADAPTIVE RATE WIRELESS COMMUNICATION SYSTEMS	**5. FUNDING NUMBERS**
6. AUTHOR(S) Peter M. B. Harley	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

In this dissertation, we investigated the methods for development of embedded channels within error correction mechanisms utilized to support adaptive rate communication systems. We developed an error correction code-based embedding scheme suitable for application in modern wireless data communication standards. We specifically implemented the scheme for both low-density parity check block codes and binary convolutional codes. While error correction code-based information hiding has been previously presented in literature, we sought to take advantage of the fact that these wireless systems have the ability to change their modulation and coding rates in response to changing channel conditions. We utilized this functionality to incorporate knowledge of the channel state into the scheme, which led to an increase in embedding capacity. We conducted extensive simulations to establish the performance of our embedding methodologies. Results from these simulations enabled the development of models to characterize the behavior of the embedded channels and identify sources of distortion in the underlying communication system. Finally, we developed expressions to define limitations on the capacity of these channels subject to a variety of constraints, including the selected modulation type and coding rate of the communication system, the current channel state, and the specific embedding implementation.

14. SUBJECT TERMS covert communications, information hiding, forward error correction, modulation and coding schemes, adaptive rate wireless communication systems, wireless local area networks, embedding, IEEE 802.11ad, directional multi-Gigabit (DMG), IEEE 802.11ac, very high throughput (VHT), low-density parity-check codes, convolutional codes	**15. NUMBER OF PAGES** 167
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

THIS PAGE INTENTIONALLY LEFT BLANK

ERROR CORRECTION CODE-BASED EMBEDDING IN ADAPTIVE RATE WIRELESS COMMUNICATION SYSTEMS

Peter M. B. Harley
Commander, United States Navy
BS, U.S. Naval Academy, 2003
MS, Naval Postgraduate School, 2010

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2019

Approved by:

John C. McEachen
Department of Electrical
and Computer Engineering
Dissertation Supervisor and Chair

Murali Tummala
Department of Electrical and
Computer Engineering
Dissertation Supervisor

David R. Canright
Department of
Applied Mathematics

Frank E. Kragh
Department of Electrical
and Computer Engineering

John D. Roth
Department of Electronic Systems
Engineering and Applied
Mathematic Programs Office

Approved by:

Douglas J. Fouts
Chair, Department of Electrical and Computer Engineering

Orrin D. Moses
Vice Provost of Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

In this dissertation, we investigated the methods for development of embedded channels within error correction mechanisms utilized to support adaptive rate communication systems. We developed an error correction code-based embedding scheme suitable for application in modern wireless data communication standards. We specifically implemented the scheme for both low-density parity check block codes and binary convolutional codes. While error correction code-based information hiding has been previously presented in literature, we sought to take advantage of the fact that these wireless systems have the ability to change their modulation and coding rates in response to changing channel conditions. We utilized this functionality to incorporate knowledge of the channel state into the scheme, which led to an increase in embedding capacity. We conducted extensive simulations to establish the performance of our embedding methodologies. Results from these simulations enabled the development of models to characterize the behavior of the embedded channels and identify sources of distortion in the underlying communication system. Finally, we developed expressions to define limitations on the capacity of these channels subject to a variety of constraints, including the selected modulation type and coding rate of the communication system, the current channel state, and the specific embedding implementation.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

List of Acronyms and Abbreviations

A-MPDU	aggregate medium access control (MAC) Protocol Data Unit (MPDU)
AL-FEC	application layer-forward error correction (FEC)
ARQ	automatic repeat request
ASCII	American Standard Code for Information Interchange
AWGN	additive white Gaussian noise
BCC	binary convolutional code
BCJR	Bahl-Cocke-Jelinek-Raviv
BER	bit error ratio
BPSK	binary phase-shift keying
BI	beacon interval
BW	bandwidth
C2	command and control
CBAP	contention based access period
CEF	channel estimation field
CSMA/CA	carrier-sense multiple access with collision avoidance
dB	decibel
DCF	distributed coordination function
DMG	directional multi-gigabit
DOCSIS	data over cable service interface specification

DVB-S	digital video broadcasting – satellite
EDMG	enhanced directional multi-gigabit
FEC	forward error correction
FSK	frequency-shift keying
Gbps	gigabits-per-second
GHz	gigaHertz
GI	guard interval
HT	high throughput
IEEE	Institute of Electrical and Electronics Engineers
IP	internet protocol
LDGM	low-density generator matrix
LDPC	low-density parity check
LLR	log-likelihood ratio
LOS	line-of-sight
LTE	long term evolution
MAC	medium access control
Mbps	megabits-per-second
MCS	modulation and coding scheme
MFB	modulation and coding scheme (MCS) feedback
MHz	megaHertz
MIMO	multiple-input and multiple-output
mmWave	millimeter wave

MPDU	MAC Protocol Data Unit
MRQ	MCS request
MU-MIMO	multi-user multiple-input and multiple-output (MIMO)
NLOS	non-line-of-sight
NR	new radio
ns	nanoseconds
OFDM	orthogonal frequency-division multiplexing
OSI	open systems interconnection
PER	packet error ratio
PHY	physical layer
PPDU	physical layer (PHY) protocol data unit
PSDU	PHY service data unit
PSK	phase-shift keying
QAM	quadrature amplitude modulation
QC-LDPC	quasi-cyclic LDPC
QPSK	quadrature phase-shift keying
RCPC	rate-compatible punctured codes
rms	root mean squared
RS	Reed-Solomon
SC	single carrier
SDR	software defined radio
SISO	single-input and single-output

SU-SISO	single user single-input and single-output (SISO)
SNR	signal-to-noise ratio
SP	service period
STA	station
STF	short training field
TGac	Task Group ac
TGad	Task Group ad
TGay	Task Group ay
TGn	Task Group n
URA	uniform rectangular array
VHT	very-high throughput
Wi-Fi	wireless-fidelity
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	wireless local area network

Acknowledgments

I want to express my sincere appreciation to my advisors, Professor Murali Tummala and Professor John McEachen, for their patience and guidance over the past three years. This process has been an incredible learning experience, and the greatest challenge of my academic career; thank you for keeping me on the right path.

I am also grateful to my peers in the doctoral cohort for their feedback, encouragement, and at times, commiseration. I continue to be inspired by your insightful research and look forward to crossing paths once we all make it back to the Fleet.

Finally, and most importantly, I would like to thank my wife, Erin, and my children, Eloise and Finnian, for their unwavering support and understanding during this journey. I could not have made it without you.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

Information-hiding techniques seek to exploit the unused or underutilized capacity within a given system and then pass embedded data from source to destination using available data carriers [1]. Covert channels describe communication paths that have been implemented through the use of these information-hiding techniques. As communications technology has advanced, so too has the field of information hiding. While the introduction of networked computers, digital media, and mobile communication systems have significantly increased both the opportunities and capacity for passing hidden messages, the fundamental premise of using data carriers to transport hidden messages has remained relatively unchanged. Limiting the amount of degradation caused by the embedding process minimizes the adverse impacts on the underlying data carrier and reduces the detectability of the embedded channel.

In recent years, a great deal of the focus has been on the development of covert channels within networked communication systems. Some information-hiding techniques, to include digital media steganography, exist entirely within the application layer and are agnostic to the underlying communication systems and protocols [2]. Other efforts have examined the potential to exploit available capacity across the open systems interconnection (OSI) reference model [3]. As wireless and mobile communications devices now account for a majority of all internet protocol (IP) traffic [4], there has been increased interest in the development of information-hiding techniques that exploit vulnerabilities within these systems [5]. In our work, we elected to investigate the opportunities to develop and evaluate information embedding within data carriers based on modern wireless local area network (WLAN) standards.[1]

[1] Portions of this chapter were previously published by IEEE [5]. Reprinted, with permission, from P. M. B. Harley, M. Tummala and J. C. McEachen, "High-Throughput Covert Channels in Adaptive Rate Wireless Communication Systems," *2019 International Conference on Electronics, Information, and Communication (ICEIC)*, Auckland, New Zealand, 2019, pp. 1-7. This publication is a work of the U.S. government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States. IEEE will claim and protect its copyright in international jurisdictions where permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

1.1 Motivation

Our interest in the study of embedded channels was originally focused around their potential use in cyber security applications. Information-hiding techniques utilize legitimate carriers to transport hidden messages and provide some measure of anonymity and security [5]. Recently, there has been increased recognition of the potential for these techniques to support cyber attacks and cyber crime [6]. When conducting cyber operations, the ability to deliver payloads, perform command and control (C2) of malware, and recover specified content relies upon the development of communication paths that evade cyber defenses [5]. A variety of techniques, to include anonymization and encryption, have been used extensively to support cyber operations [6], but these techniques still fail to completely mask the fact of an attack. Covert channels not only obscure the content of these vital communication links, but by also making the channels difficult to detect, they vastly complicate the defensive efforts of computer security and forensic professionals [7], [8].

In our initial survey, we found several information-hiding techniques that were able to develop low data rate covert channels that had little to no impact on the underlying communication system, including embedding in protocol headers [9] or using timing channels [10], [11]. We also found techniques that achieved very high capacities through the use of specific cover objects, but these objects were permanently distorted by the embedding process [12]–[14]. In our work, we sought to develop a technique for wireless communication networks that could deliver high data rates with low distortion.

While our focus was initially on the covert applications of information-hiding techniques, it quickly became apparent that they are widely utilized in support of a variety of legitimate applications to include watermarking digital media to assist with digital rights management (DRM) [15], network flow analysis to include tracing the source of denial-of-service attacks [16], [17], and the subversion of attempted internet censorship [18], [19]. One of the most unique uses of information-hiding techniques involves the provisioning of security and authentication features to applications or protocols that did not originally include that type of protection [20].

1.2 Research Objective

In this dissertation, we focused on exploiting redundant capacity available in wireless networks to support a high-throughput embedded communications channel. We developed an error correction code-based embedding scheme suitable for application in adaptive data rate communication systems, such as those based on modern IEEE WLAN standards. We specifically implemented the scheme for low-density parity check (LDPC) block codes as well as a widely utilized binary convolutional code (BCC). While error correction code-based information hiding has been previously presented in literature, we sought to take advantage of the fact that these standards have the ability to change their modulation and coding rates in response to changing channel conditions. We utilized this functionality to incorporate knowledge of the channel state into the scheme, which led to an increase in the potential embedding capacity.

The core of our work focused on the implementation and testing of these embedding techniques via a software-enabled simulation testbed. Once the embedding techniques had been fully implemented, we conducted extensive simulations to investigate the performance of our novel approach while also developing models and analytical expressions to characterize their behavior and limitations.

1.3 Related Work

Our initial research into covert communication methods took a broad look at the field to include traditional stenographic techniques, covert channels developed at various layers of existing communications protocols, and even out-of-band communications, which are often utilized to bridge air-gapped computer networks by exploiting thermal, acoustic, and optical signatures. From the outset, we have viewed covert communications as a channel embedded within an existing communication system or data carrier. This perspective developed into two primary insights that have focused our view of the research area. The first is that the embedding of a hidden payload in an existing communications channel is analogous to the addition of another noise source. The second is that the capacity of the covert channel is limited by the amount of distortion that the underlying communication system can accept.

Based on these observations, there are two methods that would enable the underlying communication system to carry covert data; we could either increase the capacity of the channel, thereby providing additional space for the embedded information, or decrease the robustness of the channel, utilizing some of that available capacity to store the covert payload. The following areas of related work supported our development of error correction code-based embedding in wireless communication systems.

1.3.1 Steganography

One of the earliest terms used to describe information hiding was steganography, which encompasses a wide range of techniques in which a secret message is embedded into a data carrier or a cover object in such a way that the modification is not noticeable [21]. The first information-hiding techniques sought to conceal the presence of communications in a manner that avoided detection by human senses to include sight, touch, or hearing [1].

A general observation from even the earliest information-hiding techniques is that the selection of the data carrier was as important as the information-hiding methodology. The more data an object is able to accept without revealing the presence of the secret message, the more effective it will be as a data carrier. In addition, it is also important that the chosen data carrier be common enough to not be considered abnormal [1]. In a steganographic channel, there already exists a non-deterministic cover source that has some measure of probabilistic variability [22]. Possible covers with this type of random distribution include digital media, the payload of packets that contain normal network traffic, or physical layer transmissions that traverse a noisy channel [22].

In modern networked communication systems, a popular high-throughput information-hiding technique is digital media steganography in which a cover object is subsequently embedded with a designated payload in such a manner that is imperceptible to both unwitting users and potential eavesdroppers [23]. The cover objects employed in digital media steganography are considered to be expendable, and therefore, the permanent degradation of the object is not considered to be a negative attribute. While these techniques can deliver high throughput with low distortion, there are some drawbacks that could limit their utility in support of cyber operations. Specifically, it is necessary to select an appropriate cover object that must be tolerant of the embedding distortion without exhibiting any obvious changes. Accordingly, the selection of cover objects for digital-media steganography is normally limited to images, video, or audio and requires the generation of specific traffic from the source computer, which may not conform to expected network activity.

1.3.2 Covert Channels in Wireless Networks

With the popularity of wireless networks, there has been extensive research conducted into information-hiding techniques that exploit these protocols, resulting in a wide range of proposed methods that implement both storage and timing channels [9], [24]–[26]. Additionally, there have been other interesting applications that attempted to exploit the physical layer (PHY) characteristics of the wireless channel and conduct information hiding in a manner that did not significantly degrade channel performance.

One of the most comprehensive discussions of PHY information hiding is discussed in [27], where they developed, implemented, and tested wireless covert channels under the 802.11a/g standard. This effort was particularly notable as it utilized a software defined radio (SDR) to conduct real-world testing of their techniques as opposed to purely relying on simulation. Four specific covert channels were evaluated including phase-shift keying (PSK) modulation within the short training field (STF), frequency-shift keying (FSK) modulation within the carrier frequency offset, transmission of additional subcarriers, and embedding in the cyclic prefix. Another notable example explored two methods of developing covert channels within the Worldwide Interoperability for Microwave Access (WiMAX) standard [28]. The first derived capacity from frame padding while the second, a forward error correction (FEC)-based implementation, leveraged the error correction capacity of a Reed-Solomon (RS) code to carry a limited embedded payload. Finally, [29] looked at the development of a covert channel in the 802.11n standard; this implementation also relied on modifications to the cyclic prefix of orthogonal frequency-division multiplexing (OFDM) symbols. Of note, this technique advertised the highest covert channel capacity of any of these PHY techniques with a data rate of up to 19.5 megabits-per-second (Mbps).

1.3.3 Application of Error Correction Coding in Information Hiding

The use of error correction codes in information hiding can be divided into three focus areas: to protect the embedded information, to support digital media steganography, and to enable physical-layer embedding of hidden data. The most common application of error correction codes is to protect the hidden data being transmitted across the covert channel. As in standard communication systems, the use of these techniques is governed by channel conditions and the desire to have the hidden data arrive without error [15]. In some respects, an information-hiding implementation may place an even greater emphasis on the need for the delivery of error-free data than a normal communication channel; depending on the specific implementation, it may not be possible for the receiving station to request retransmission for lost or corrupted data [22].

Another popular application for error correction codes in information hiding is digital media steganography; specifically, the use of structures initially developed for error correction in a technique known as matrix embedding [23]. Matrix embedding utilizes the properties of random linear block codes to embed hidden data in objects before transmission; specific codes utilized in matrix embedding steganography include simplex codes (which are duals of Hamming codes) [23], [30], low-density generator matrix (LDGM) codes (which are duals of LDPC codes) [31], and quasi-cyclic LDPC (QC-LDPC) codes [32].

The final application exploits the functionality of FEC codes to carry hidden information and develop covert communication channels [28], [33], [34]. FEC is an attractive avenue for information hiding as these codes often provide more redundancy than required by channel conditions; this redundancy can be used to carry hidden data [5]. Additionally, most modern communication protocols also include retransmission mechanisms that can resend lost or corrupted data if the embedded FEC fails to correct all bit errors [34] [5]. Unlike traditional stenographic schemes in which the cover object is irreparably degraded by the insertion of the hidden payload, FEC-based embedding schemes can avoid permanent corruption of the legitimate data. Despite there being numerous examples of error correction code-based embedding, there was no recognition of the potential to exploit the relationship between the selection of alternate modulation and coding rates and increased capacity.

1.3.4 Error Correction Code Embedding

Many information-hiding techniques are focused on simply developing proof-of-concept implementations. That said, during our initial investigation of information-hiding techniques, we were able to gain some valuable insight into previous analysis conducted on notional error correction code-based information hiding that helped establish the behavior of these techniques.

Yan et al. focused on the derivation of capacity for an error correction code-based information-hiding scheme; specifically, they completed a theoretical derivation of channel capacity based on both an ideal state where all errors were corrected and a non-ideal state where errors remained after decoding [33]. From these generalized cases, the authors made some initial observations including that the maximum size of the hidden message increases as the channel noise decreases, the length of the source data increases, and the error-correcting capability of the selected FEC increases. While these results completely align with our intuition on the subject, they also helped to inform the development of our initial embedding methodology, which sought to utilize information about the current channel state to increase the embedding capacity without introducing additional uncorrectable errors in the underlying communication system.

Safir et al. examined stenographic embedding at the physical layer of a baseband communication channel [35]. Similar to our initial insights on embedding in error correction codes, this work concluded that by controlling the number of embedding modifications, this form of steganography could be accomplished without permanent degradation of the underlying message. Additionally, there was a focus on determining the ability of an eavesdropper to distinguish the noise component due to the embedding of data from the normal channel noise. These observations related directly to the distortion considerations of our proposed channel and the recognition that the channel estimation methods utilized by the underlying communication system could potentially identify the distortion caused by our techniques.

1.3.5 Desirable Features of Information-Hiding Techniques

In considering the development of a new embedding methodology, we also gained valuable insights into relevant characteristics of information-hiding techniques. A comprehensive set of information-hiding attributes was initially described in [15], and although the focus of their work was watermarking, the general insights on restrictions and features remain relevant to modern information-hiding schemes. The work stressed the importance that the selected embedding technique should minimize any degradation on the underlying data carrier to ensure that the presence of the hidden data remains undetected, even if it is visible upon close inspection [15]. There was also a significant focus on ensuring the integrity of the hidden data. Specifically, [15] addressed the use of error correction codes, making the embedded data resistant to manipulation and ensuring that the data would be recoverable if only a fragment of the host signal was available. This final feature seemed applicable when looking at the segmentation of data before it is transmitted over a wireless network, and how to minimize the impact on the embedded channel in the event of the loss or corruption of a frame and its associated embedded payload.

An alternate view of these design considerations was provided by the efforts in [22] to develop a comprehensive taxonomy for covert channels. The most significant considerations from this work related to the need to define the level of exploit required to implement the proposed channel and whether the channel is likely to support bidirectional communication. While the level of exploit does not directly influence the capacity of the proposed channel, it does speak to the level of difficulty in implementing a particular information-hiding technique and whether physical access or hardware modifications are required. The second consideration, which is in some ways tied to the required level of exploit, is critical in determining the level of error correction and protection that must be applied to the underlying embedded payload. If the proposed implementation has the ability to support duplex signaling, then a variety of options exist to recover a corrupted payload by employing techniques similar to automatic repeat request (ARQ). On the other hand, if only simplex communication is likely, then the aggressive use of FEC techniques are required; these codes will greatly improve reliability at the cost of throughput.

In this section, we examined information-hiding research which was relevant to our objective. We identified shortcomings of previous work with respect to low data rates, restrictive selection of data carriers, or permanent distortion of the underlying communications channel. Our work focused on maximizing the potential capacity of an error correction-based embedding technique for adaptive rate wireless communication systems; these techniques do not require the selection of a particular data carrier and minimize the possibility of permanently altering the underlying payload.

1.4 Outline

This dissertation is organized as follows. Chapter 2 provides background information on topics related to modern wireless-fidelity (Wi-Fi) standards. Concepts for the development of embedded channels in adaptive rate wireless communication systems are then presented in Chapter 3. The specific implementation of the embedding concepts for LDPC and convolutional codes are presented in Chapters 4 and 5, respectively. Results from the embedding simulations for LDPC and convolutional codes are presented in Chapter 6 under both additive white Gaussian noise (AWGN) and fading channel conditions. Significant contributions from this research, and potential future work, are presented in Chapter 7. Finally, a sample of the code utilized to implement embedding simulations is provided in the Appendix.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Modern Wireless Communication Systems

The focus of our research was the development and analysis of embedding techniques within the error correction codes of modern wireless communication systems. The purpose of this chapter is to provide an overview on some of the fundamental concepts that support our work.[2]

2.1 Modulation and Coding Schemes

All modern IEEE 802.11 protocols, as well as other modern wireless communication systems including long term evolution (LTE), rely on a predetermined set of modulation types and coding rates to facilitate communication across a range of channel conditions [36], [37]. These modulation and coding schemes (MCSs) are assigned an index value to enable easy coordination between the transmitting and receiving station. The flexibility of MCS-based systems enable wireless networks to optimize throughput while maintaining an acceptable error performance [36].

The range of performance offered by these systems is well illustrated through the results of a PER simulation for the 802.11ad directional multi-gigabit (DMG) single carrier (SC) PHY as shown in Figure 2.1. In this standard, a total of 19 separate MCS not only allow the system to maintain communications over a wide range of channel conditions, but also support multiple data rates. A critical function of MCS-based systems is the ability to transition between MCS indices in response to changing channel conditions; link adaptation is the process that allows stations to select the optimal MCS for the current channel state. There is no standard implementation for link adaptation, but packet error ratio (PER) is a common metric utilized to characterize the performance of a channel, and therefore evaluate the selection of an MCS [36]. PER represents a measure of performance based on the number of packets that are not correctly received after transmission. The

[2] Portions of this chapter were previously published by IEEE [5]. Reprinted, with permission, from P. M. B. Harley, M. Tummala and J. C. McEachen, “High-Throughput Covert Channels in Adaptive Rate Wireless Communication Systems,” *2019 International Conference on Electronics, Information, and Communication (ICEIC)*, Auckland, New Zealand, 2019, pp. 1-7.

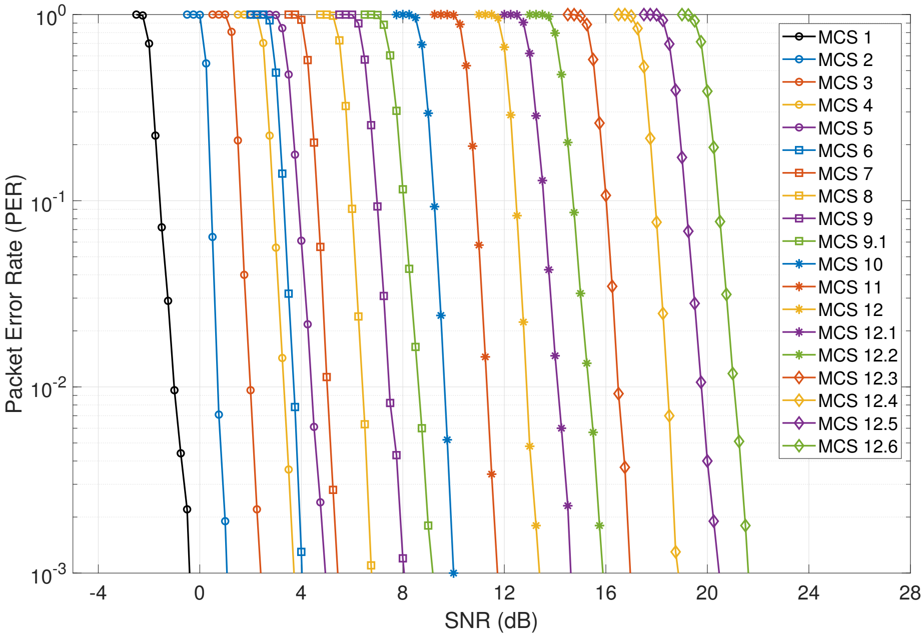


Figure 2.1. Packet error ratio versus SNR performance curves for Single Carrier IEEE 802.11ad DMG modulation and coding schemes; performance average of 10000 4096-octet PSDU over AWGN channel.

selection of the MCS by the link adaptation scheme is intended to maximize throughput while maintaining PER below an acceptable limit. One method for selecting the MCS is to simply measure the PER over time and adjust the selected MCS value to keep errors below a prescribed threshold. A more dynamic approach, which would achieve higher throughput, is to dynamically select an MCS that is optimized for changes in the channel condition.

If we generalize rate adaptation algorithms, they can be separated into two categories, loss-based algorithms or signal-to-noise ratio (SNR)-based algorithms [38]. For the purpose of our research, we were primarily interested in the viability and evaluation of SNR-based approaches [39]–[43]. While these efforts have validated that SNR is a good indicator of channel quality, they have identified a number of challenges including asymmetric performance in the channel [43], the need to account for differences in SNR thresholds due to the use of uncalibrated equipment, and the existence of other interference that can increase the predicted PER [42]. Despite these challenges, it has been observed that SNR-based algorithms can provide performance improvements over traditional loss-based algorithms [40], [42].

Recent WLAN standards have recognized the need for transmitting and receiving stations to coordinate in the link adaptation process. In 802.11n, this functionality was formalized through the inclusion of a Link Adaptation Control field as part of the HT Control field; 802.11ac performs a similar function using the very-high throughput (VHT) variant of the HT Control field with the contents of the MCS request (MRQ) and MCS feedback (MFB) fields. In 802.11ad DMG, link adaptation is accomplished through the use of Link Measurement Request and Link Measurement Report frames [44]. These mechanisms allow the receiving station to monitor the received transmissions and make reports to the transmitter that can be used to select the appropriate MCS for the current channel state. Information about the quality of the channel for this MCS recommendation can be derived from measurements made by the receiving station itself as well as information about the reliability of received transmissions [36]. The channel measurements made at the receiver are likely to exploit the ability of modern communication systems to conduct accurate channel estimation. This channel estimation is critical for the proper operation of these communication systems as it is required to assist with synchronization, measure fading, and also provide a noise variance estimate that is utilized to perform soft decision demodulation [45].

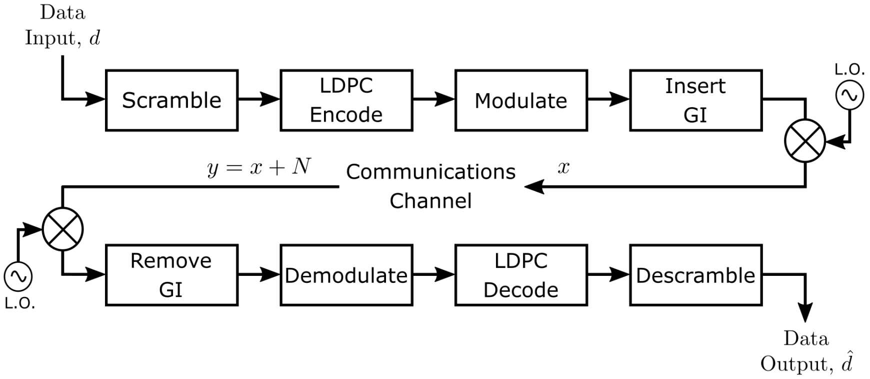
2.2 Modern Wireless Local Area Network Standards

While the embedding methodology presented in this dissertation could be applied to a wide range of error correction coding applications, we specifically focused on two of the most recent Wi-Fi standards, 802.11ac VHT, and 802.11ad DMG. While these standards have been optimized for very different use cases, they both have the potential to achieve extremely high data rates in excess of 1 gigabits-per-second (Gbps); as a result, both standards could support extremely high throughput embedded channels.

2.2.1 Directional Multi-Gigabit (DMG), IEEE 802.11ad

The IEEE 802.11ad PHY and medium access control (MAC) amendment for millimeter wave (mmWave) WLANs was formally adopted as the DMG specification in 2012. While IEEE 802.11ad operates in one of six 2.16-gigaHertz (GHz) channels in the 60 GHz range, DMG is a term that is applied to any WLAN operating in channels with a starting frequency above 45 GHz [44]. The combination of high frequency and high channel bandwidth allows

DMG to achieve data rates in excess of 8 Gbps [46]. The standard originally specified three different PHY modes: control, SC, and OFDM [5]. The control and SC PHY are mandatory for all devices with an optional low-power SC PHY for power-constrained applications; the OFDM PHY is now obsolete [46]. Baseband processing for all SC PHY involves a scrambler, encoder, modulator and insertion of guard intervals (GIs) [5]. A simplified block diagram of the transmit and receive process is shown in Figure 2.2 [5].



Original figure adapted from [46].

Figure 2.2. DMG transmit and receive process. Source: [5], © 2019 IEEE.

IEEE 802.11ad is intended for use in high-bandwidth, short-range, line-of-sight (LOS) applications to include wireless cable replacement for high definition video [46], wireless peripherals and docking stations [47], or other traditional WLAN implementations [5]. The low latency and characteristics of the DMG PHY, have led to the development of additional use cases to include wireless virtual reality hardware [48] and a variety of mobile communication and sensing applications [49]. Development of a follow-on mmWave WLAN implementation continues through the IEEE P802.11 Task Group ay (TGay); enhanced directional multi-gigabit (EDMG) will remain backward compatible with DMG, but will utilize channel bonding and aggregation to achieve even higher data rates [50]. Potential EDMG applications include IEEE 802.3 Ethernet replacement or high-capacity backhauls for data centers and telecommunications [50].

The MAC layer of DMG also marks a departure from previous Wi-Fi protocols. The primary difference with the DMG MAC is that it does not solely rely upon the distributed coordination function (DCF) present in other IEEE 802.11 implementations. This change is primarily due to the unique constraints of mmWave and the fact that traditional carrier-sense multiple access with collision avoidance (CSMA/CA) is difficult in DMG due to higher attenuation and the use of highly directional links [51]. The use of beamforming techniques on these links improve SNR and allow for spatial reuse of frequency bands [47]. In lieu of traditional DCF, DMG relies upon a combination of scheduled access, similar in implementation to time division multiple access (TDMA), while retaining some elements of CSMA/CA [52]. The primary method for a DMG station (STA) to gain access to the communication medium is through the use of a scheduled service period (SP). A diagram of the DMG MAC architecture is shown in Figure 2.3a with a representation of DMG channel access in Figure 2.3b. The contention based access periods (CBAPs), which can be used to provide access for stations that do not have an assigned SP, does support CSMA/CA but prevents participating stations from utilizing beamforming. The use of an SP is the preferred method for gaining channel access, while CBAPs are generally used by STA to request SP assignments [51].

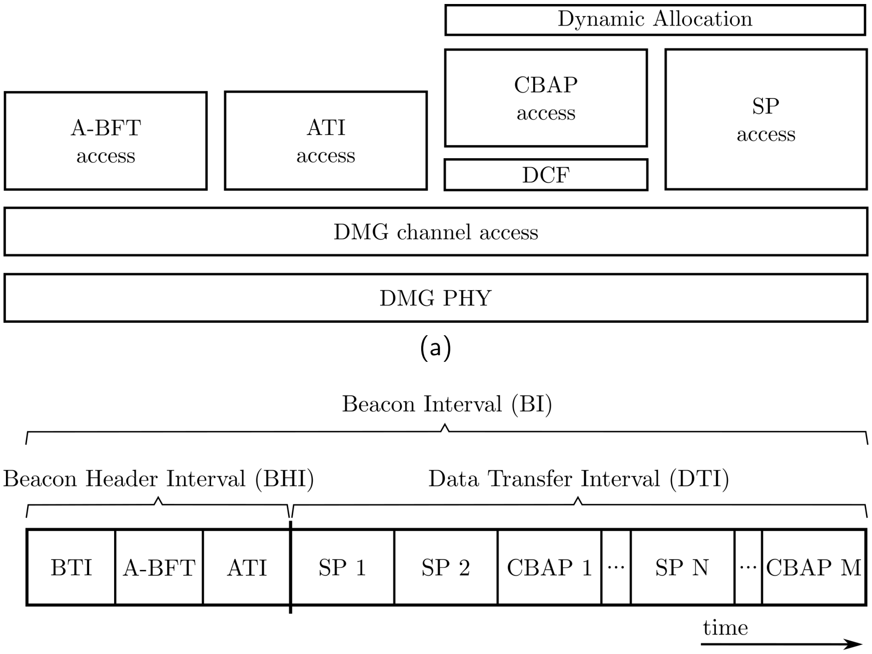


Figure 2.3. DMG MAC detail: (a) block diagram of MAC architecture and (b) channel access and detail of beacon interval. Adapted from [44].

2.2.2 Very-High Throughput (VHT), IEEE 802.11ac

The second modern Wi-Fi standard we have elected to investigate is IEEE 802.11ac VHT. In many ways, VHT is simply an extension of the existing 802.11n high throughput (HT) standard with enhancements that were intended to deliver wireless gigabit network speeds. The key attributes of the VHT specification that account for this improved performance are increased channel bandwidth, increased number of multiple-input and multiple-output (MIMO) spatial streams, downlink multi-user MIMO, and the addition of 256-quadrature amplitude modulation (QAM). The VHT specification was formally adopted in 2012 and operates in 20, 40, 80, or 160 megaHertz (MHz) channels in the 5 GHz band [44]; it also supports up to eight spatial streams as well as 8×8 MIMO.

Beyond the PHY enhancements, the remainder of the VHT specification is quite similar to that of HT. One major shift involved the assignment of MCS index values. In HT, MCS values were assigned for all of the different combinations of modulation type, coding rate, spatial streams and channel bandwidth; for VHT this has been greatly simplified with only ten MCS indices (0-9). The VHT protocol supports both a BCC and a LDPC code; for the purposes of this research, we have focused only on the use of the BCC in VHT, which is part of the mandatory protocol implementation [44]. The BCC utilized by this scheme is a constraint length, $K = 7$, with generator polynomials $g_0 = 133$ and $g_1 = 171$; we will also use the notation $(7, [133, 171])$ to reference this specific convolutional code. All available code rates utilized by VHT are developed by puncturing the output from the $1/2$ rate BCC encoder.

2.3 Relevant Error Correction Codes

Error correction coding is an essential component of modern communication protocols. These techniques provide error detection and correction capabilities which minimize the retransmission of data, facilitating increased throughput and reduced latency. Our research into embedding within error correction codes focused on both LDPC and convolutional codes.

2.3.1 Low-Density Parity Check Codes

LDPC codes are a class of channel capacity approaching codes first discovered by Gallager in the early 1960s [53]. Although these codes have achieved broad application in a wide variety of modern digital communication systems including Wi-Fi [44], digital video broadcasting – satellite (DVB-S) [54], data over cable service interface specification (DOCSIS) 3.1 [55], and 5G new radio (NR) [56], they were not fully explored for more than 40 years; the slow adoption was mainly due to the computational complexity required for implementation, and the fact that the algebraic block codes and convolutional codes in use during the 1960s were more than sufficient to meet demand [57]. Interest in LDPC codes was revived following the discovery of turbo codes in the early 1990's. Due to the proprietary nature of the published turbo codes, researchers returned to Gallager's original work [58]. This coding technique, which could now be supported by modern computing resources, delivered extremely high performance but had significantly lower decoding complexity than turbo codes [57].

LDPC codes are linear block codes, that can be fully specified by either their generator matrix, $\boldsymbol{G}$, or parity-check matrix $\boldsymbol{H}$. A resulting LDPC codeword, $\boldsymbol{v}$, of length q, must satisfy the parity-check equations specified by $\boldsymbol{H}$ such that [59]

$$\boldsymbol{v}\boldsymbol{H}^T = \mathbf{0} \ . \tag{2.1}$$

The unique structure of LDPC codes was originally defined so that each row and column of $\boldsymbol{H}$ contained a small fixed number of 1's [60]; it has been found that irregular codes, where the number of 1's in each column can vary, outperform the original regular code specification [58]. LDPC codes are often visualized through the use a bipartite graph introduced in [61], a construct now known as a Tanner graph. This representation is particularly useful when thinking about the LDPC decode process as it clearly identifies the relationship between the message (or variable) nodes and their associated check nodes. A simplified LDPC parity check matrix and associated Tanner graph are shown in Figure 2.4; in the Tanner graph, the message nodes, c_i, are represented by circles while the check nodes, f_i, are represented by squares. The message nodes represent all of the codeword symbols from the LDPC code and include both the original data as well as the parity bits that were appended during the encode process. The relationship between the parity check matrix and Tanner graph in Figure 2.4 is represented by the color-coded edges, or paths, that connect the message and check nodes.

$$H = \begin{bmatrix}
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & \textcircled{1} & 0 & 0 & \textcircled{1} & 0 & \textcircled{1} & 0 & 0 & 0 & \textcircled{1} & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
\textcircled{1} & 0 & 0 & 0 & 0 & 0 & \textcircled{1} & 0 & 0 & 0 & 0 & 0 & 0 & \textcircled{1} & 0 & \textcircled{1} \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

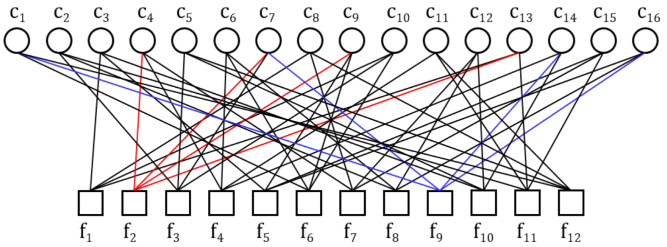


Figure 2.4. Notional LDPC code parity check matrix and associated Tanner graph. Adapted from [58].

LDPC codes are known as sparse graph codes; they are sparse due to the relatively small number of edges when compared to what would be expected for a fully connected graph. As a result, decoding complexity remains linear even as the size of the parity check matrix increases [62]. In the example matrix provided in Figure 2.4, each column only contains three 1's; the fact that the weight of each column is the same means that this would be considered to be a regular LDPC matrix. Unlike the syndrome decoding utilized for simpler linear block codes, LDPC codes utilize iterative message passing algorithms [58]. In the most basic hard-decision decoding example, check nodes receive inputs from all of their connected message nodes and then decide whether any of the bits are in error based on whether the message bit contributions pass the appropriate parity check [58]. The check node then returns this feedback to the message nodes; the message nodes combine all of the received check node feedback with the received message bit to determine if the received bit value was in error. This process can then be continued until either the parity check equation is satisfied or the maximum number of message passing iterations is reached [62].

2.3.2 Convolutional Codes

Convolutional codes were first discovered by Elias in 1955 [57]. Unlike the block codes previously discussed in this section, the convolutional encoder contains 'memory' and therefore the encoded bit stream depends not only on the current input but also previous inputs [57]. As a result, the coder for a rate $R = k/n$ convolutional code with a memory of

M can be realized utilizing a k-input, n-output linear sequential circuit [59]; although there is often competing nomenclature in literature, the memory of the code, M, is often specified as a constraint length, K, where $K = M + 1$. The constraint length represents the number of total bits that contribute to the output of the encoder; in the case of the primary BCC examined in this dissertation, the constraint length of $K = 7$ means that a total of seven bits, the current input bit plus the six previous inputs stored in memory, all contribute to the determination of the output. An example 1/2-rate BCC encoder is shown in Figure 2.5.

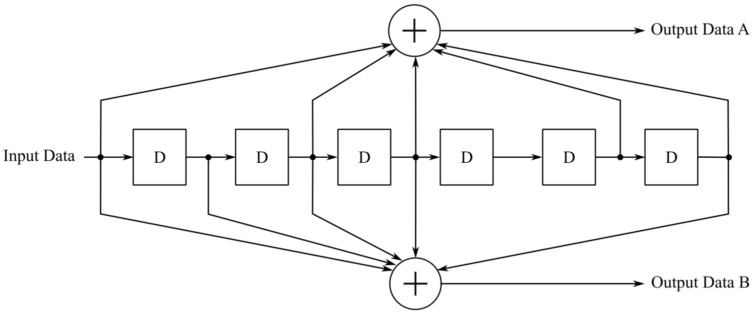


Figure 2.5. Rate 1/2 convolutional encoder specified by generators $g_0 = 133$ and $g_1 = 171$, with constraint length, $K = 7$. Adapted from [44].

Important contributions to develop efficient decoding methods for convolutional codes include work by Wozencraft and Reiffen, Massey, Viterbi and Bahl-Cocke-Jelinek-Raviv (BCJR) [57]. Applications of convolutional codes include satellite communication, radio, mobile communications, and digital video. A common technique to extend the performance of a parent convolutional code is to use puncturing techniques to create a series of higher rate codes. The puncture process for a $R = 1/2$ parent code is illustrated in Figure 2.6.

One of the features that makes punctured convolutional codes attractive is that the same encoders and decoders can be utilized for both the parent and punctured codes. While the same decoder structure is used, methods must be employed to handle the bit locations that have been punctured. An example of the trellis of a punctured code is shown in Figure 2.7, where the punctured bit locations are indicated by an “X.” These puncture locations are handled by stuffing dummy bits into the punctured locations at the receiver [53], [63] or by simply ignoring the puncture bit locations in the decoder [59].

A variation on puncturing that influenced the development of our convolutional code

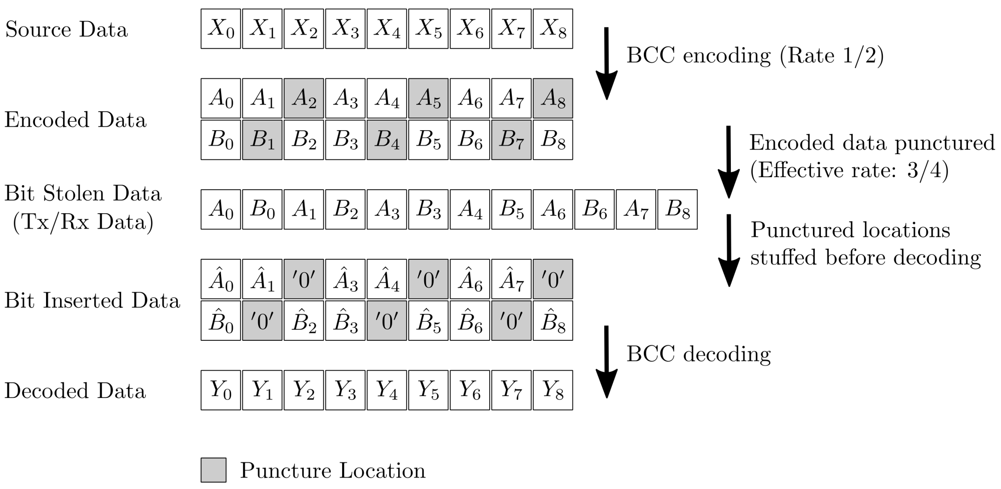

Source Data
X_0 X_1 X_2 X_3 X_4 X_5 X_6 X_7 X_8
BCC encoding (Rate 1/2)
Encoded Data
A_0 A_1 A_2 A_3 A_4 A_5 A_6 A_7 A_8
B_0 B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8
Encoded data punctured
(Effective rate: 3/4)
Bit Stolen Data
(Tx/Rx Data)
A_0 B_0 A_1 B_2 A_3 B_3 A_4 B_5 A_6 B_6 A_7 B_8
Punctured locations
stuffed before decoding
Bit Inserted Data
$\hat{A}_0$ $\hat{A}_1$ '0' $\hat{A}_3$ $\hat{A}_4$ '0' $\hat{A}_6$ $\hat{A}_7$ '0'
$\hat{B}_0$ '0' $\hat{B}_2$ $\hat{B}_3$ '0' $\hat{B}_5$ $\hat{B}_6$ '0' $\hat{B}_8$
BCC decoding
Decoded Data
Y_0 Y_1 Y_2 Y_3 Y_4 Y_5 Y_6 Y_7 Y_8
Puncture Location

Figure 2.6. Puncturing process for convolutional codes. Adapted from [44].

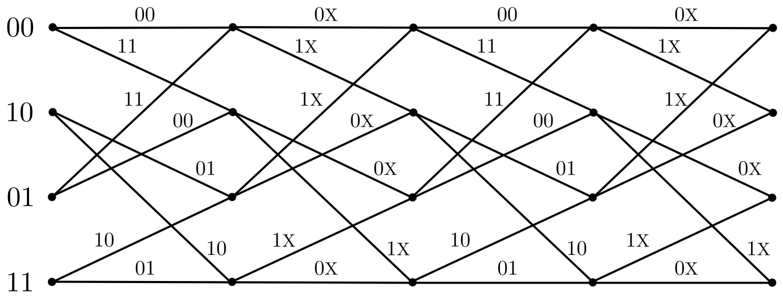


Figure 2.7. Decode process for $R = 2/3$ punctured convolutional code with generators $g_0 = 3$ and $g_1 = 7$. Adapted from [53].

embedding technique was the concept of rate-compatible punctured codes (RCPC). First proposed in [64], RCPC are utilized in applications where unequal levels of error protection are required within a particular information block. As a result, their unique structure enables the rapid switching between code rates within a single frame. To accommodate this type of application, the development of these codes require a specific search technique; starting with the highest rate code, all of the subsequent lower rate codes must utilize the same bit locations while including one or more additional parity bits [65].

Traceback Depth

In order to achieve the maximum performance for the Viterbi decoding of a convolutional code, it is necessary to maintain a sufficient path history [66]. In practical application, however, there is a desire to minimize the storage requirements for the path histories while maintaining sufficient information to achieve good decoding performance. This limit, or traceback depth, τ, describes the number of trellis stages that are stored within memory.

For unpunctured codes, a traceback depth of five times the constraint length is generally considered adequate [63]; for punctured codes, the traceback depth must be increased and the extent of this increase is generally determined experimentally for each puncturing rate and pattern. Alternatively, there has been work presented to provide an estimate for the required τ for various puncturing rates based on both the code rate itself and the encoder memory length, M; an expression to estimate the required traceback depth, $\hat{\tau}$, was presented in [66]

$$\hat{\tau} \geq \frac{5M}{2\,(1-R)} = \frac{5\,(K-1)}{2\,(1-R)}, \tag{2.2}$$

where R is the rate of the code. The memory of the encoder can also be defined in terms of the constraint length, K, with $M = K - 1$. This estimate is utilized within MATLAB to determine the τ utilized by the implementation of the soft-decision Viterbi decoder for the BCC; the resulting traceback depth requirements are provided in Table 2.1 for each BCC code rate.

Table 2.1. Traceback depth utilized in MATLAB for IEEE 802.11ac BCC decoder, K=7

BCC Code Rate	**Traceback Depth, τ**
$R = 1/2$	30
$R = 2/3$	45
$R = 3/4$	60
$R = 5/6$	90

The selection of τ is important to our area of embedding research because our proposed embedding mechanisms mimic the impact of puncturing on a received code; specifically, we are removing the information contained in these bit locations and thereby reducing the information available to make path selections.

2.4 Log-Likelihood Decoding of Low-Density Parity Check Codes

Although hard decision decoding is often utilized in academic environments, in practice soft-decision decoding is utilized to improve the performance of the communications channel. The use of log-likelihood ratio (LLR) is an attractive option for real-world soft decision demodulators because it provides similar performance as soft decision probabilities, but can be represented as a fixed point value without the need to accept floating-point representations [53]. The following section provides a brief description of LLR as well as describes LDPC decoding using LLR values.

2.4.1 Log-Likelihood Ratio

The LLR output values of the receiving station demodulator represent both a bit value and a confidence level. In general terms, the LLR of $\tilde{x}$ is a ratio of probabilities [53]

$$\lambda\left(\tilde{x}\right) = \log\left[\frac{P\left(\tilde{x} = 1\right)}{P\left(\tilde{x} = -1\right)}\right], \tag{2.3}$$

where log is a natural logarithm. To interpret the meaning of an LLR value, we can take the example of representing the probability that $\tilde{x} = 1$ in terms of $\lambda\left(\tilde{x}\right)$ [53]

$$P\left(\tilde{x} = 1\right) = \frac{e^{\lambda(\tilde{x})}}{1 + e^{\lambda(\tilde{x})}}. \tag{2.4}$$

We find that the sign of the LLR provides an indication of the bit value, while the magnitude, $|\lambda\left(\tilde{x}\right)|$, provides an indication of the reliability [53]. This measure of reliability is of particular importance to our proposed embedding scheme; to facilitate high rates of embedding in systems that utilize LLR demodulated values, it necessary to set these values to zero at the embedding locations. Similar to an erasure, an LLR = 0 represents a level of zero confidence (or complete uncertainly) and therefore these bit locations are less detrimental to the decode process. In the MATLAB implementations described in Chapters 4 and 5, the LLR values are the result of an approximation [67]. This approximation, which only accounts for the nearest symbol location as opposed to the distance from all symbols, has been found to significantly reduce processing requirements without significantly impacting performance [68].

2.4.2 Log-Likelihood LDPC Decoding

As described in our earlier example, the iterative decoding for LDPC codes is a series of updates performed to the message and check nodes based on the propagation of beliefs, or probabilities, across the edges described in the Tanner graph [59]. This process can also be performed in a similar manner using LLR values in lieu of hard decision values or probabilities. Starting with the LLR values received from the demodulator, $\lambda(c_n|\boldsymbol{r})$, which are based on the channel reliability, the iterative decode process conducts a series of check node and bit node updates. First, the check node updates are performed using [53]

$$\eta_{m,n}^{[l]} = -2\tanh^{-1}\left(\prod_{j\in\mathcal{N}_{m,n}} \tanh\left(-\frac{\lambda_j^{[l-1]} - \eta_{m,j}^{[l-1]}}{2}\right)\right), \tag{2.5}$$

where l is the loop counter, which begins with $l = 1$ on the first cycle, $\lambda_j^{[l-1]}$ are all of the contributing LLR values for check node m where $j \neq n$, and $\eta_{m,n}^{[l]}$ is the check node update, for positions where $\boldsymbol{H}(m,n) = 1$. For the first cycle, all positions in $\eta_{m,n}^{[0]}$ are initialized at 0. These check node updates are then utilized to compute bit node updates [53]

$$\lambda_n^{[l]} = L_c r_n + \sum_{m\in\mathcal{M}_n} \eta_{m,n}^{[l]}, \tag{2.6}$$

where r_n is the received bit probability, and L_c is the channel reliability. Beyond describing the behavior of the LLR-based LDPC decoders that are utilized in our simulated results, this algorithm provides insight into how the LDPC decoder will deal with the introduction of LLR = 0 into the representation of the received message bits, $\lambda^{[0]}$. It becomes clear that in the first computation of the check node update using (2.5), any $\lambda_n^{[0]} = 0$, which indicates the presence of a puncture (or embedded) bit location, $\eta_{m,j}^{[1]} = 0$ where $j \neq n$. This mechanism effectively prevents the LLR = 0 from corrupting the other message bits that participate in the impacted check nodes. After the bit update at the end of the first cycle, however, the punctured message location, $\lambda_p^{[1]}$, will have obtained a value estimate from the input provided by the check nodes, and consequently will participate in the next cycle of the check node update.

2.5 Summary

In this chapter, we presented relevant foundational concepts related to modern wireless communications systems that influenced our research. Understanding key characteristics of these specific WLAN standards, and details of their error correction codes, provided sufficient insight to propose the framework for our embedding methodologies.

CHAPTER 3: Embedded Channels in Adaptive Rate Communication Systems

In this chapter, we outline our development of the error correction code-based embedding technique that is at the core of our research. Although heavily based on previous information-hiding techniques, we chose to classify our research as the development of an embedded channel; this classification was due to the fact that these channels have other potential applications beyond covert communications, and the fact that our research did not explicitly evaluate the covert nature of our proposed scheme.

Our research has been focused on wireless communications, and in particular current IEEE 802.11 WLAN protocols. This focus is not only due to the ubiquitous nature of these systems, but also the fact that they possess characteristics, like variable error correction coding rates and multiple modulation types, that we exploited to develop high capacity embedded channels.[3]

3.1 Embedded Channels in Communication Systems

In [1], the authors present a characterization of information hiding when viewed against the backdrop of all possible methods to conceal data in communication networks. As shown in Figure 3.1, information hiding makes up just one of three concealment techniques. Distinct from anonymity techniques that obscure the identities of the parties involved in the communication exchange or cryptographic techniques that protect the communication content from unauthorized disclosure, information hiding is concerned with concealing the communications process itself.

Under this rubric, our proposed embedding methodology, which would be invisible to the legitimate users of the communication system under most conditions, is most

[3] Portions of this chapter were previously published by IEEE [5]. Reprinted, with permission, from P. M. B. Harley, M. Tummala and J. C. McEachen, "High-Throughput Covert Channels in Adaptive Rate Wireless Communication Systems," *2019 International Conference on Electronics, Information, and Communication (ICEIC)*, Auckland, New Zealand, 2019, pp. 1-7.

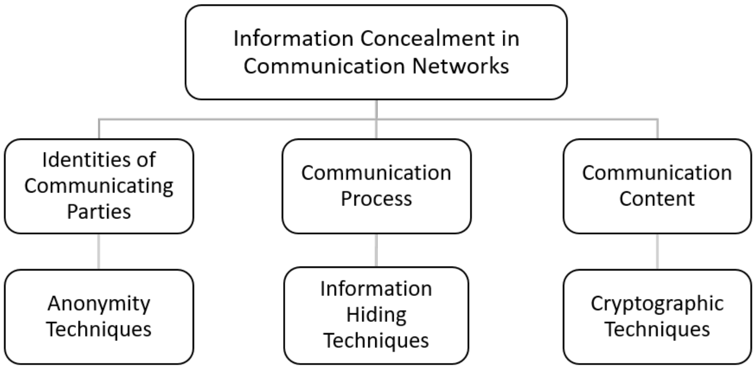

Information Concealment in Communication Networks
Identities of Communicating Parties
Communication Process
Communication Content
Anonymity Techniques
Information Hiding Techniques
Cryptographic Techniques

Figure 3.1. Classification of information hiding in communication networks. Adapted from [1].

closely aligned with information hiding. We began our conceptual development of embedding in wireless communication systems by considering the simple implementation of an information-hiding technique in a notional communication channel. The performance of a communications system under normal conditions, D_{Norm}, measured in terms of data rate, is shown in Figure 3.2. We then established an upper and lower bound, identified as the upper rate limit, D_U, and the lower rate limit, D_L. This system is ultimately constrained by Shannon capacity, C, which establishes the maximum theoretical capacity for the channel [5].

The upper rate limit, D_U, represents the case where technical measures, possibly an alternate error correction code, increase the potential data rate of the communication system; this increased performance could then be utilized to support a covert channel with a maximum data rate represented by the difference between curves D_U and D_{Norm} [5]. Alternatively, D_L represents a situation where the performance of the communication system is intentionally degraded; this lower-bound would be further limited by the threshold of minimum acceptable performance, D_{Req}, or other detection considerations [5]. In this case, the capacity of the covert channel is the difference between the normal operation, D_{Norm}, and either the degraded condition, D_L, or other required performance bounds [5]. In both cases, the difference between D_{Norm} and D_U, D_L, or D_{Req} represents distortion. Minimizing distortion will reduce the impact on the underlying system while limiting detectability of the covert channel. In the next section, we will discuss a scheme that leverages the capacity

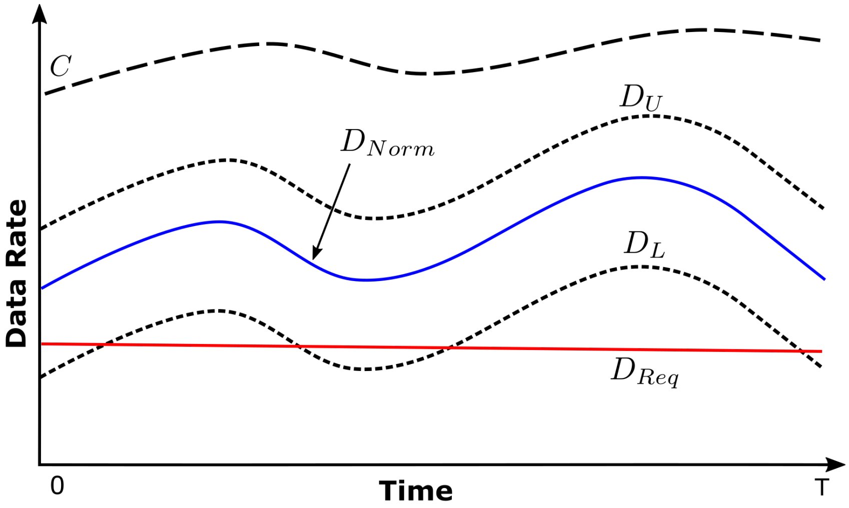


Figure 3.2. Generalized information hiding model of a communication system as it relates to the Shannon capacity as well as the normal operation of the system

between the D_{Norm} and D_L curves [5].

3.2 Preliminary Insight

Our preliminary insights into our proposed embedding technique were gained during a review of the DMG PHY specification within the IEEE 802.11-2016 standard; this review was intended to identify any attributes that could be readily exploited for the development of a covert communications channel.

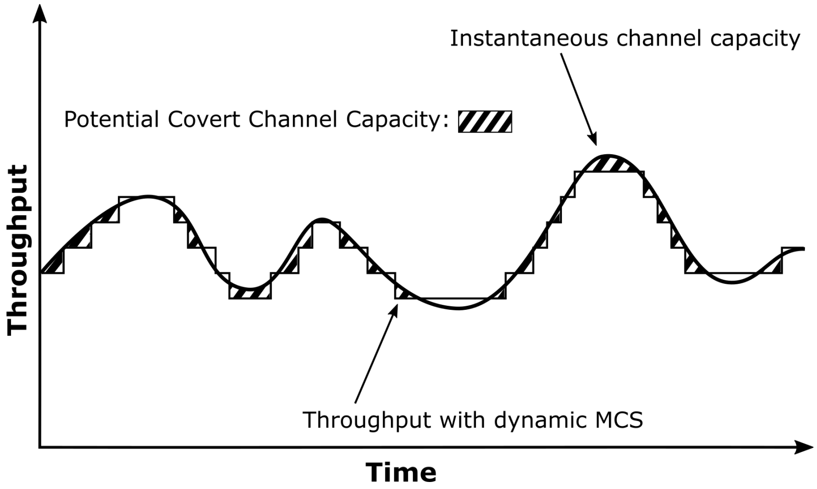
We identified a number of potential techniques, which ranged from simple bit stuffing in unused header fields, to dirty constellation coding of transmitted symbols, to even possible exploitation of the Golay sequences that were utilized in the STF and channel estimation field (CEF) of the PHY header to facilitate synchronization and channel estimation. The most intriguing opportunity, however, was related to a revision that had been made to the DMG SC PHY when it was determined that the OFDM PHY was obsolete [46]. To increase the maximum data rate of the SC PHY, seven additional MCS indices (9.1 and 12.1 - 12.6) were added to the specification. The higher data rates were achieved through the introduction of a new modulation type, 64-QAM, and by the implementation of a 7/8-rate LDPC code. The characteristics of the SC MCS indices, to include error correction coding scheme and rate, modulation type, number of bits per symbol, N_{CBPS}, repetition factor, ρ, and the maximum data rate are contained in Table 3.1.

Table 3.1. Summary of MCS for SC IEEE 802.11ad DMG. Adapted from [44], [46].

MCS Index	Modulation	N_{CBPS}	LDPC Rate	Repetition	Data Rate (Mbps)
1	$\pi/2$-BPSK	1	1/2	2	385
2			1/2	1	770
3			5/8		962.5
4			3/4		1155
5			13/16		1251.25
6	$\pi/2$-QPSK	2	1/2		1540
7			5/8		1925
8			3/4		2310
9			13/16		2502.5
9.1			7/8		2695
10	$\pi/2$-16QAM	4	1/2		3080
11			5/8		3850
12			3/4		4620
12.1			13/16		5005
12.2			7/8		5390
12.3	$\pi/2$-64QAM	6	5/8		5775
12.4			3/4		6390
12.5			13/16		7507.5
12.6			7/8		8085

Although distinct $\boldsymbol{H}$ were specified for all of the existing DMG LDPC rates, the new 7/8-rate code was achieved by passing data through the existing 13/16-rate encoder and then puncturing the first 48 parity bits [5]. We theorized that if an SC DMG system was operating under channel conditions that supported a 7/8-rate code MCS (i.e., MCS 9.1), the 48 parity bits that would normally be punctured might be able to carry an embedded payload if a user intentionally selected an MCS that utilized the 13/16-rate code (i.e., MCS 9) [5]. Our novel approach would therefore leverage the MCS construct and link adaptation functionality to increase the potential embedding capacity of a technique similar to those used in FEC-based information hiding.

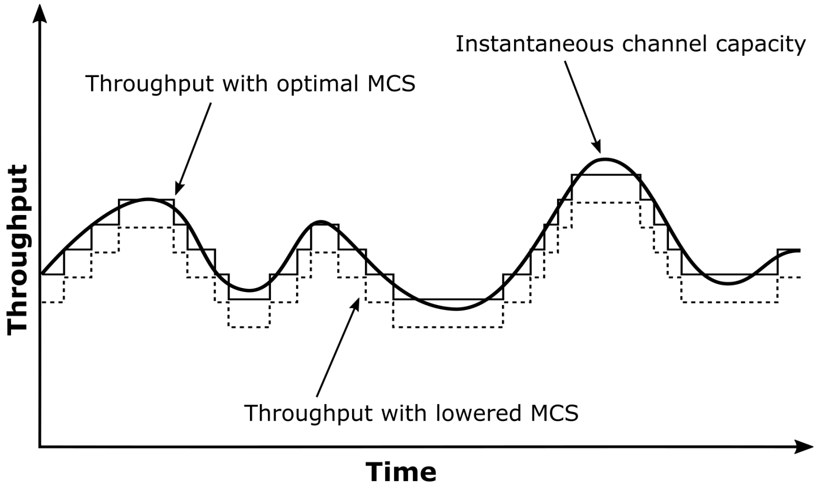
Under normal operation with an MCS determined by the channel state, the payload of a FEC-based information-hiding scheme can be represented by the shaded area of Figure 3.3, where the embedded channel capacity is the difference between the two curves.


Instantaneous channel capacity
Potential Covert Channel Capacity: ▨
Throughput
Throughput with dynamic MCS
Time

Original figure adapted from [36].

Figure 3.3. Baseline covert channel capacity of modulation and coding scheme-based information-hiding technique. Source: [5], © 2019 IEEE.

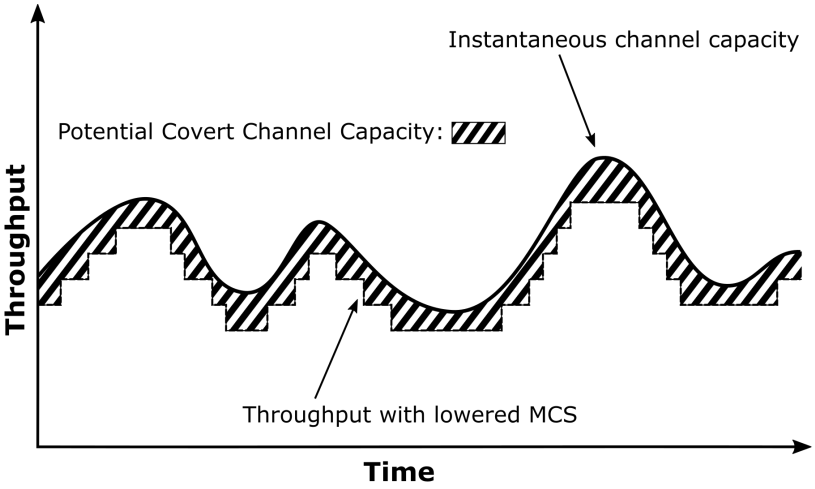
Attempting to increase the throughput of the information-hiding scheme will exceed the instantaneous channel capacity and result in data from the underlying system being delivered with an increased probability of error [5]. Our proposal would first change the performance of the underlying communication channel as shown in Figure 3.4 through the intentional selection of a lower MCS index.


Instantaneous channel capacity
Throughput with optimal MCS
Throughput
Throughput with lowered MCS
Time

Original figure adapted from [36].

Figure 3.4. Throughput of communication system with lower MCS intentionally selected. Source: [5], © 2019 IEEE.

This action would increase the embedded channel capacity, illustrated in Figure 3.5, without causing increased error rates to the underlying communications channel.



Original figure adapted from [36].

Figure 3.5. Increased covert channel capacity of modulation and coding scheme-based information-hiding technique with lower MCS selected. Source: [5], © 2019 IEEE.

Since our proposed method involved an embedding technique that is similar to puncturing, implementation would require access to both the transmitting and receiving station. The specific modifications to the transmitter and receiver were not rigorously explored in our work, but both stations would likely require changes to the firmware and/or hardware.

3.3 MCS-based Embedding Considerations

Traditional error correction based information-hiding implementations exploit the availability of excess error correction capacity to support the embedding of hidden data. In this work, we found that we were able to demonstrate significantly higher embedding capacity by making two reasonable assumptions.

The first assumption is that we would acquire sufficient access to the PHY to facilitate the removal of the embedded data from the received codeword before decoding. In our view, this is reasonable as punctured bit locations must be dealt with similarly as to not interfere with the decoding process. We would use these existing mechanisms to exclude the values of our embedded bit locations from the decoding process, and as a result, each embedded bit would have significantly less impact on the underlying channel. While these bit locations would not contribute information to the decoding process, they also would not impart additional errors to the legitimate bits.

The second assumption is that our embedding scheme would have some level of insight into the current channel conditions. This was also deemed reasonable. Modern wireless communication systems already maintain some sense of the channel state through their normal operation. This insight can either be explicitly determined using channel estimation as a means to assist with decoding, synchronization or link adaptation, or implicitly through the selection of an MCS index.

With these considerations in mind, we investigated two distinct modes in this work. The first relies upon decrementing the MCS, while the second attempts to utilize existing excess capacity to support the embedded channel. We also briefly discuss some of the considerations associated with the use of FEC techniques to ensure reliable delivery of our embedded payload.

3.3.1 Decremented MCS Embedding Implementation

In this first case, we propose intentionally selecting the next lower MCS index, or decrementing the MCS, in order to increase the effective redundancy of the error correction code. Through this process, we obtain useful information about the channel state without having to explicitly take measurements. If the channel was supporting a specific MCS index, it would certainly support the next lower MCS index, which by design provides more redundancy at the expense of data rate.

We then propose using this extra redundancy to support our hidden channel. Because of the known relationship between these two MCSs, we also assert that it would be possible to characterize the amount of data we could embed while maintaining the same channel error performance, measured in PER. Due to these known characteristics, this proposed implementation would only be required to pre-coordinate the specific location of the embedding, and the number of bits that would be embedded in a given codeword or frame. As shown in Figure 3.6, an embedding control block would consider both the current MCS, as well as any MCS feedback provided within the protocol standard, before selecting a decremented MCS index along with the appropriate embedding rate. This fixed embedding rate would be predetermined based on the known differences between adjacent MCS indices.

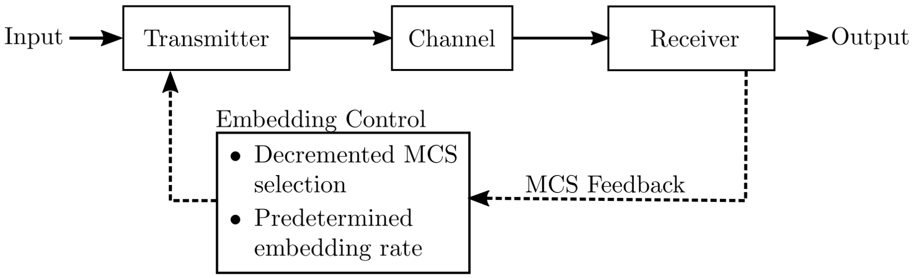


Figure 3.6. Block diagram of notional decremented MCS implementation

One major issue with this approach, particularly in terms of any potential covert application, is that it lowers the data rate of the underlying communication channel. While this reduction may not be apparent to the user without detailed information about current channel conditions, it is a potentially noticeable performance impact.

3.3.2 Variable Rate Embedding

The second proposed implementation is considerably more complex from both a channel estimation standpoint but also in the coordination of the embedding location. Although MCS indices cover a range of operating conditions, there are channel states that fall between adjacent indices. We would aim to leverage previous research into channel estimation and SNR-based link adaptation to exploit these regions to support an embedded channel. Not only would this scheme require an accurate estimation of the margin between the current channel conditions and the minimum required to support the desired performance at the current MCS index, but also a method to equate that channel margin into an embedding capacity. A simplified block diagram representing this implementation is shown in Figure 3.7. Furthermore, successful extraction of the embedded bits prior to decoding would require coordination of the resulting embedding rate and location with the receiver.

Implementation in conjunction with the decremented MCS approach described above would require the development of decision logic at the transmitter to determine whether the available channel state could support the embedded channel requirements or whether it would be necessary to decrement the MCS index to facilitate a higher throughput. While

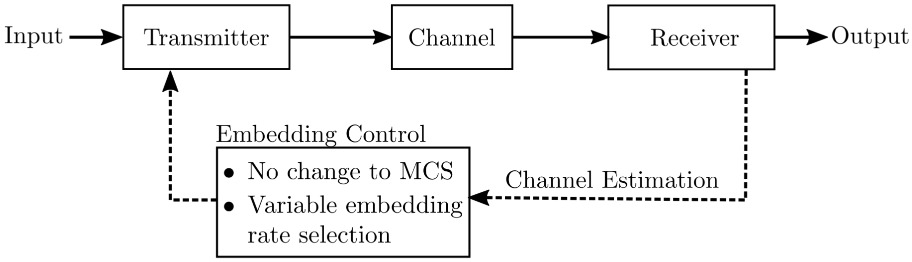


Figure 3.7. Block diagram of notional variable embedding implementation

this implementation is considerably more complex, it would allow the development of the embedded channel without significantly impacting the data rate of the underlying system.

3.3.3 Error Protection of Embedded Payload

The final consideration of implementing this embedding scheme is ensuring that the embedded payload can be reliably delivered. In evaluating options for information-hiding techniques, it is important to determine whether the proposed channel will support unidirectional (simplex) or bidirectional communications. In a full or half-duplex channel, errors can be corrected through the use of retransmission protocols; if the channel only supports simplex communication, the embedding scheme must rely entirely on FEC mechanisms to correct errors encountered in the channel [22]. Given the challenges expected with implementing this type of embedded channel, we must assume the worst case of simplex-only communication.

As a result, the selection of an appropriate error correction mechanism is critical to channel performance. In general, if we are embedding in an underlying channel with coding rate R, we can select the same rate for our embedded data, a rate that provides more redundancy than the underlying channel at the expense of data rate, or a higher rate code that provides less redundancy than the underlying channel but increases the embedding payload. In the case where we have intentionally decremented the MCS of the underlying communication system to support our embedded channel, we could comfortably select a higher rate FEC code based on our knowledge that the channel is better than implied by the MCS index in use.

3.4 Adaptive Rate Embedded Channel Model

If we consider a general MCS-based communication system, each modulation and coding rate pair support a different data rate. A description of the available channel throughput is characterized by the relationship between the current channel condition and the number of errors observed at the receiver. An important component to MCS-based systems is the ability to estimate the current channel conditions and select an appropriate MCS index. While specific rate-adaptation implementations vary, in most cases, the selection of an MCS acts like a floor-function; the channel state may exceed the minimum SNR required to support a given MCS index, but the system must select that lower rate to maintain the desired error performance. This error threshold is often defined as a PER for a given length PHY service data unit (PSDU). For 802.11ad, this performance threshold is considered to be 1% PER for a 4096-octet PSDU [44] [5].

We developed a simplified model to better understand the relationship between the selected MCS, the current channel state, and the available capacity that could be utilized to support FEC-based embedding. In Figure 3.8, the curves labeled I_{C-1}, I_C, and I_{C+1} represent the PER performance of three MCS indices across a range of SNR values. The MCS index, I_C, is selected based on the current channel state and resulting signal-to-noise ratio, SNR_{curr}. In our scenario, the available SNR exceeds the minimum requirements of the communication system, and as a result, some of the error correction capability provided by the selected FEC is redundant; this excess capability could then be utilized to support the embedding of information within the FEC codewords.

The embedding capacity of each codeword could be increased so long as the PER performance of the communication system remained at or below the specified protocol threshold; this threshold is designated as PER_T and represents the maximum acceptable PER for a given length PSDU. At maximum embedding capacity, the performance of the communication system would be represented by the dashed curve I_{C_E}. This curve would intersect the specified performance threshold, PER_T at SNR_{curr}. The difference between the SNR value at this intersection and the SNR required to maintain PER_T with an unembedded MCS is defined as the embedding margin, labeled M_E.

The proposed model also provides some insight into both the capacity of our notional embedded channel as well as a measure of distortion that occurs as a result of the embedding

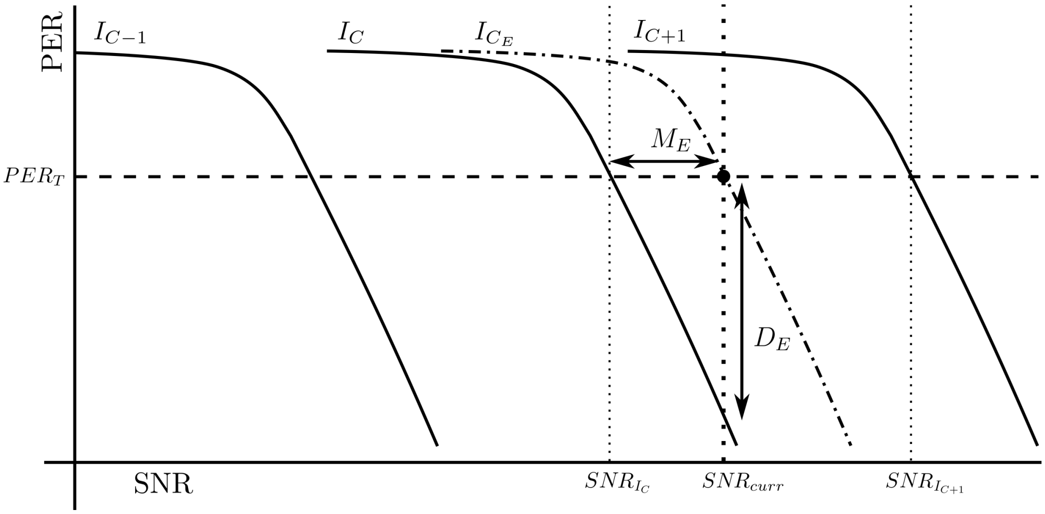


Figure 3.8. Rate adaptive embedding model, packet error ratio versus SNR

implementation. The distortion, labeled as D_E, represents the difference between the expected PER for a given MCS at the current channel state and PSDU size, and the PER observed when the channel is embedded; this distortion has a range between zero and the threshold limit PER_T. We also proposed a lower bound for the embedding capacity, C_{CW}, in bits per codeword, where r_C and r_{C+1} are the number of redundant bits for the MCS indices C and $C+1$ respectively, and SNR_{I_C} and $SNR_{I_{C+1}}$ are the SNR required to maintain the specified PER error threshold for those same MCS indices. Assuming that the available space was fully embedded, or $M_E = SNR_{I_{C+1}} - SNR_{I_C}$, then we would expect that C_{CW} would be greater than or equal to the difference in redundancy between the two MCS indices

$$C_{CW} \geq \frac{M_E\left(r_C - r_{C+1}\right)}{SNR_{I_{C+1}} - SNR_{I_C}} \ . \tag{3.1}$$

If we look more broadly at our FEC-based embedded channel implementation, it is possible to propose an absolute upper bound for the resulting covert channel based on Shannon capacity, C. The capacity limit for the underlying communication system can be determined as [69]

$$C = W \log_2 \left(1 + \frac{S}{N}\right) \tag{3.2}$$

where W is the channel bandwidth, S is the signal power, and N is the noise power. Since our FEC-based scheme occupies a known proportion of the total number of coded bits transmitted by the underlying communication system, it is reasonable to assume that these bits occupy an equivalent component of the total bandwidth. If we define this fractional component, μ, as a ratio of the number of embedded bits to the total number of bits in the coded bitstream, it is possible to rewrite (3.2) as

$$C_E \leq \mu W \log_2 \left(1 + \frac{S}{N}\right) \tag{3.3}$$

to specify an upper limit of our embedding capacity, C_E, for our current channel state. Since C for a given SNR is finite, the resulting capacity of the legitimate communications channel, C_L would be reduced by an equivalent amount

$$C_L = C - C_E \leq W\left(1 - \mu\right) \log_2 \left(1 + \frac{S}{N}\right) . \tag{3.4}$$

3.5 Summary

In this chapter, we presented our proposed embedding implementation for adaptive rate communication systems. We explored some of the assumptions that were required to support the increased data rate provided by this proposed method as well as presented two distinct modes of implementation. Finally, we presented a model of our proposed embedding methodology as it relates to the operation of an adaptive rate MCS-based communication system.

CHAPTER 4: Error Correction-Based Embedding in Low-Density Parity Check Codes

Our initial efforts to conduct embedding in an adaptive rate wireless communication system focused on IEEE 802.11ad DMG. As discussed in Chapter 3, we elected to investigate two potential embedding methodologies during our research. The first method would exploit the multiple MCS indices in the adaptive rate communication system by intentionally decrementing the selected MCS index prior to embedding. The second implementation, which did not impact the MCS selection, was developed to utilize available information about the current channel state to exploit the additional redundancy within the existing MCS. Our development of embedding in the LDPC codes utilized by 802.11ad evolved from simple proof-of-concept demonstrations. We gradually increased the complexity, examining the application of forward error correction to the embedded payload, investigating multiple embedding locations within the LDPC codewords, developing analytical tools and methods, and extending our simulations to validate performance under multipath fading channels.[4,5]

4.1 Embedding with IEEE 802.11ad Directional Multi-Gigabit WLAN

The core functionality of our proposed embedding technique depends upon the ability to embed hidden data within the error correction mechanisms of an MCS-based wireless communication system. Data embedding occurs at baseband after the LDPC encoder. Due to the proposed implementation, embedding locations must be coordinated between the transmitter and receiver through a pre-shared key. The embedding location can be selected at any bit location; to align with the method used to implement the 7/8-rate puncturing scheme, the initial location selected to embed the payload bits was in the first *n* parity bits of

[4] Portions of this chapter were previously published by IEEE [5]. Reprinted, with permission, from P. M. B. Harley, M. Tummala and J. C. McEachen, "High-Throughput Covert Channels in Adaptive Rate Wireless Communication Systems," *2019 International Conference on Electronics, Information, and Communication (ICEIC)*, Auckland, New Zealand, 2019, pp. 1-7.

[5] Portions of this chapter were used in an upcoming paper submission for the 53rd Hawaii International Conference on System Sciences (HICSS), slated for January, 2020.

each codeword. Once the embedding process was complete, the modified LDPC codeword was passed to the modulator before completing the rest of the transmit process. At the destination, the embedded message was recovered after demodulation [5].

The output of the receiving station demodulator are LLR values. After extraction of the embedded message, the bit positions that carried the hidden data are assigned a value of 0 before being sent to the LDPC decoder. In [44], it specifies that for punctured codes, LLR values of 0 are used at the decoder to prevent the stuffed bits from introducing unnecessary errors in the decoding process; this recommendation aligns with our analysis on how the LLR-based LDPC decoder handles 0 values during the first iteration of the message passing algorithm. This same principle is leveraged to prevent the LLR values of the embedded hidden data from corrupting the legitimate payload. A block diagram of the information hiding architecture, including the proposed use of FEC to protect the hidden data, is shown in Figure 4.1.

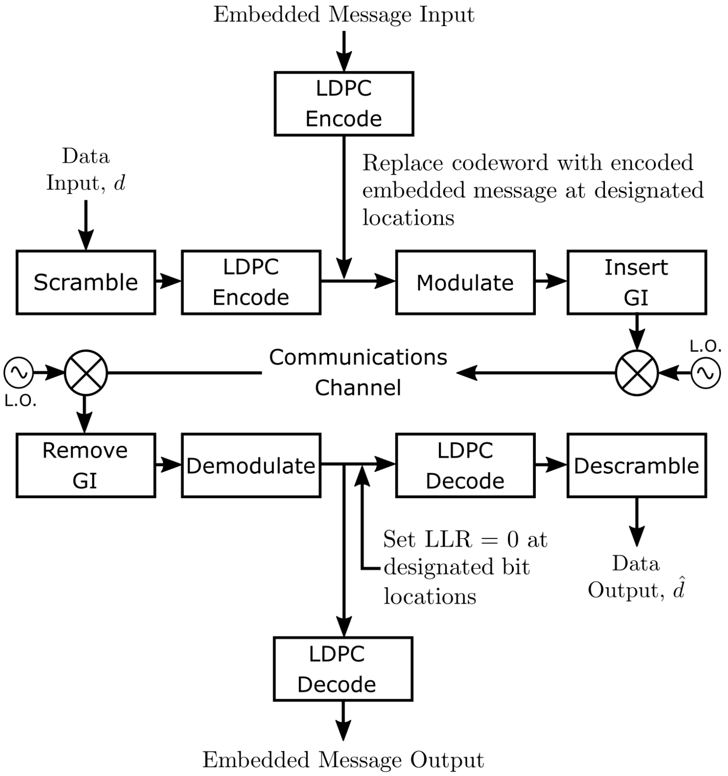


Figure 4.1. Major components of the embedding process within 802.11ad SC PHY. Adapted from [46].

The LDPC codes utilized in DMG are systematic and the embedded bits were initially inserted into the first n parity bits of the LDPC codewords. This embedding location was initially selected as it mirrored the puncturing location specified in [44] to generate the $R = 7/8$ code needed to support MCS 9.1, 12.3 and 12.6. As the mechanism utilized to embed data in FEC codewords mimics the method used for puncturing, it is reasonable to assume that optimized puncturing locations would yield the highest possible embedding rates. Although many methods exist to develop optimized puncturing patterns through the application of algorithms or computer search [70]–[73], devoting significant time to this effort was beyond the scope of our work.

We did, however, conduct embedding trials at multiple locations within the LDPC codewords in an attempt to identify alternate locations that may yield improved payload capacity over the original embedding locations. Specifically, we explored embedding locations at the beginning, middle, and end of the data and parity bit sections of the LDPC codeword. Despite the fact that these specific codes are systematic, the method for decoding LDPC means that it does not expressly matter if data bits or parity bits are punctured (or embedded). The most critical factor appears to be related to the column weight, w_c, of $\boldsymbol{H}$ in the specified embedding locations.

The importance of w_c in the selection of the embedding locations is most likely due to the observations of the iterative decoding process for LDPC with LLR inputs presented in Section 2.4.2. With an LLR = 0, the punctured (or embedded) bit location relies upon the aggregated inputs from the other received symbols to recover the transmitted value; w_c indicates the number of message bits participating in the decode process at each check node [59]. To maximize the amount of information which contributes to the recovery of the data lost during puncturing, or the transmission of an embedded payload, locations should be selected that maximize w_c.

Although we did not conduct an exhaustive search, we determined that embedding in the last n data bits of each codeword generally returned a performance gain over embedding in the first n parity bits. The only exception to this trend was for embedding in the already punctured $R = 7/8$ code; in this case the original location returned better performance, with higher embedding rates achieved at lower PER. If we examine the selected embedding locations in terms of w_c, we do find that our observations on column weight are supported by the simulated results. The average column weight, $\bar{w}_c$, of the embedded bit locations for each of the LDPC code rates is shown in Table 4.1. The number of embedding locations participating in the average, n, is based on the maximum embedding capacity observed for the first n parity bits in the quadrature phase-shift keying (QPSK) MCS.

Table 4.1. Average column weight for embedding locations within 802.11ad LDPC codewords

LDPC Code Rate	***n***	**Last *n* Data Bits**	**First *n* Parity Bits**
		Avg Column Weight, $\bar{w}_c$	Avg Column Weight, $\bar{w}_c$
$R = 1/2$	95	4	3.4421
$R = 5/6$	85	3.506	2.9882
$R = 3/4$	48	4	3
$R = 13/16$	48	3	2.875
$R = 7/8$	21	3	2

The only case where $\bar{w}_c$ did not accurately predict the embedding performance was for the $R = 7/8$ LDPC code; that said, this code is a special case as it had already lost 48 parity bits to puncturing prior to embedding. The full comparison of embedding capacity based on location is presented in Section 6.1.1.

4.2 Simulation Development

Experimental trials of this proposed technique were conducted in MATLAB. The simulation was adapted from a MATLAB-developed script to measure PER; embedding and extraction of the hidden message required modifications to existing encode and decode functions used within the MATLAB WLAN Toolbox [5]. An example of the required modifications can be found in the Appendix A.1. Our simulation implemented the transmission of a single DMG PHY protocol data unit (PPDU) which contained a 4096-octet PSDU [5]. This PSDU length was selected based on receiver sensitivity validation criteria outlined in [44], which specified that the PER for each SC MCS index be no more than 1% given a PSDU length of 4096 octets [5]. This PER value would also be extensively used in our research to establish the minimum performance threshold for our embedding scheme and enable comparison with embedded MCS indices.

In order to assess the performance of these embedding techniques, we needed to specify a payload for both the PSDU of the underlying communication system as well as a payload for the embedded data. For the 4096-octet PSDU, we generated a 128×256 pixel bitmap image in a checkerboard pattern; the black pixels were encoded as 1 and white pixels encoded as 0. We then selected text from Chapter 1 of *Alice's Adventures in Wonderland* to be used as the embedded payload for our simulations. In both cases, the selected payload facilitated rapid visual confirmation of received bit errors as we developed our initial simulations. An example of this can be seen in Figure 4.2 where bit errors in our PSDU are easily identified.

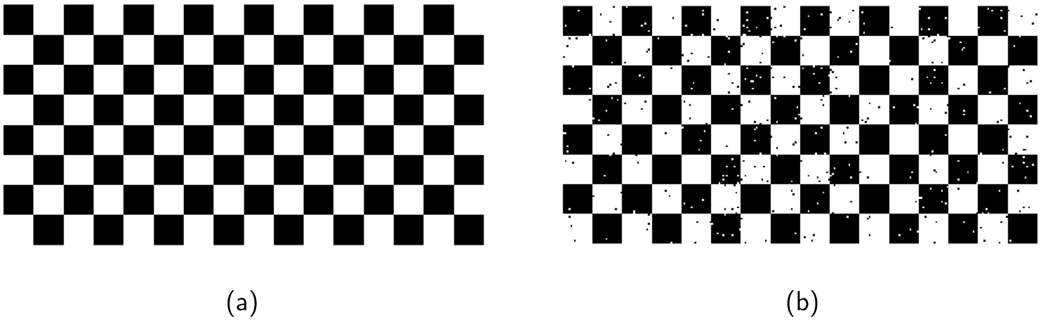


Figure 4.2. PSDU payload for simulated 802.11ad PPDU; 128×256 pixel checkerboard bitmap: (a) transmitted image and (b) received image with uncorrectable errors.

While we rapidly transitioned to other metrics to assess the performance of the embedding method, we retained the original PSDU and embedded payload for the majority of our simulated trials. As a control, we repeated our original trials for the QPSK MCS and obtained consistent results when utilizing randomly generated binary sequences for both the PSDU and the embedded payload.

All of our initial trials utilized AWGN to simulate interference in the channel. The use of a consistent seed value during AWGN generation ensured the noise environment remained consistent for each series of trials. This consistent channel allowed direct comparison of changing embedding rates on the underlying channel as the only variable impacting system performance was the amount of data embedded in each codeword.

4.2.1 Performance Metrics

Once the embedding and extraction techniques were developed and implemented in the software simulation, it became necessary to identify metrics to effectively assess the behavior and performance of the proposed methodology. Keeping in mind the information-hiding features proposed by [15], and previously discussed in Section 1.3.5, it became clear that in addition to collecting information about the embedding capacity, we also needed the ability to characterize the reliability of the embedded channel as well as assess the impact of our embedding on the performance of the underlying communications system.

The performance of our proposed techniques was assessed by measuring the PER of the underlying system as well as the bit error ratio (BER) of the received embedded payload. PER is an important performance metric as uncorrectable packet errors can result in significant reduction of channel throughput due to the requirement for retransmissions. As a result, PER served as our critical indicator of the health of the underlying communication system and provided an indication of the impact of specified embedding rates at given channel conditions. The second factor in evaluating our embedding methodology was providing an estimate of expected errors in our embedded payload. As we could not assume the availability of duplex communication for our embedded channel, it was highly desirable to minimize the BER of the embedded payload. We elected not to measure embedded payload performance in terms of a block error rate due to the fact these metrics are a function of block length. Therefore, changes in the embedding rate, and the corresponding changes to the size of the embedded payload, would make it difficult to perform direct performance comparisons.

Multiple trials were conducted to develop an accurate representation for the PER and BER at each specified SNR value; a minimum of 10000 trials were conducted to evaluate MCS and embedding combinations. In some cases the number of trials was increased to 100000 PSDU per SNR point to improve the fidelity of the results.

4.2.2 Preliminary Capacity Analysis of Decremented Embedding

Our embedding methodology looked at the transmission of data within 802.11ad as it related to both the total length of the PSDU as well as the number of embedded bits contained in each LDPC codeword. The length of the PSDU, L_P, measured in octets, not only influenced PER performance of the underlying system but also determined the number of LDPC codewords, N_{CW}, available for embedding. The embedding capacity of each PSDU, C_{PSDU}, was determined by

$$C_{PSDU} = C_{CW} N_{CW} , \tag{4.1}$$

where C_{CW} is the embedding capacity of each LDPC codeword. We then determined the number of LDPC codewords, N_{CW}, for a specified length PSDU (in octets), L_P, [44]

$$N_{CW} = \left\lceil \frac{8\rho L_P}{L_{CW} R_C} \right\rceil , \tag{4.2}$$

provided we have information about the selected MCS to include the length of the LDPC codeword in bits, L_{CW}, the code rate, R_C, and the repetition factor of the code, ρ. As noted in [44], L_{CW} will be 672 for all 802.11ad SC DMG MCS except those utilizing the punctured $R = 7/8$ code; for those MCS indices, $L_{CW} = 624$. Combining (4.1) and (4.2) yields an alternate form

$$C_{PSDU} = C_{CW} \left\lceil \frac{8\rho L_P}{L_{CW} R_C} \right\rceil . \tag{4.3}$$

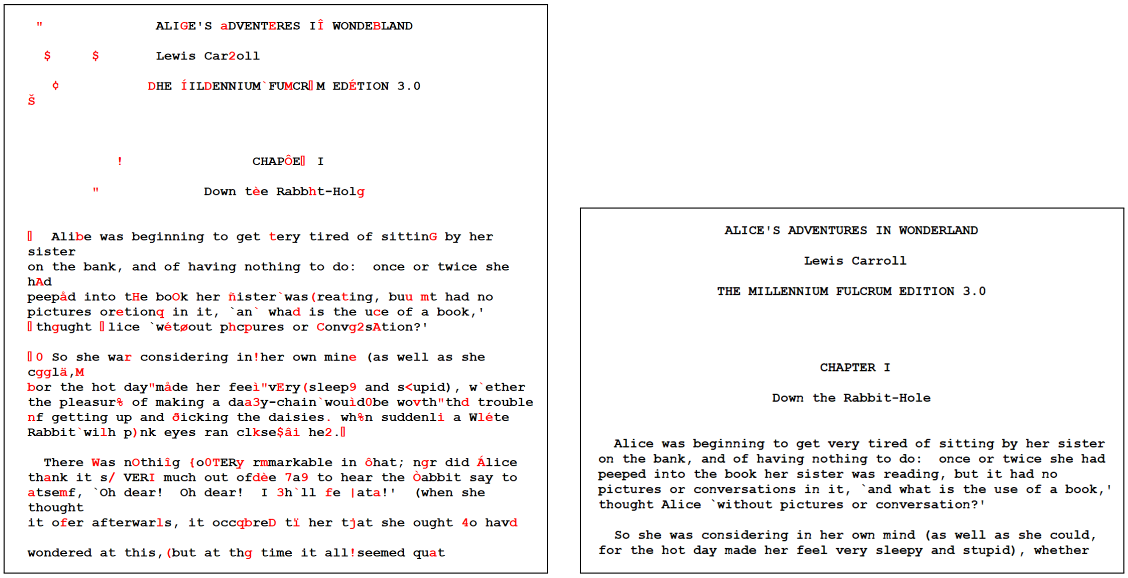
It is important to note that C_{PSDU} is only an indication of raw embedding capacity and does not address the overhead required to implement an error correction technique on the embedded payload.

4.3 Forward Error Correction of Embedded Message

Initial embedding trials did not implement an error correction scheme for the embedded payload. Instead, a simple hard decision technique was utilized on the raw LLR values; negative LLR values provided by the 802.11ad demodulator were decoded as a bit value of 1, and positive values as 0. While it was possible to make decoding decisions for each bit position, our embedding implementation does not allow us to take advantage of the inherent data protection mechanisms used on the underlying communication system.

Consequently, the error performance of our embedded payload was similar to that of an uncoded communication system operating with the same modulation type and rate.

Although implementing error correction mechanisms on our embedded data reduced the overall capacity, in practice the use of an FEC technique would be critical. The trade-off between capacity and error correction can be clearly observed in the following case where a single 4096-octet PSDU was transmitted with characteristics from MCS 6 across an AWGN channel. Before transmission, the underlying PSDU was embedded with a 9310-bit payload which contained the binary representation of 1163 ASCII characters. Upon receipt, the embedded payload was extracted and decoded per our standard embedding implementation; the underlying PSDU, which utilized a $R = 1/2$ FEC code was received without error. Unfortunately, as shown in Figure 4.3a, the uncoded embedded payload was received with a large number of uncorrectable errors which are highlighted in red.



```
 "              ALIGE'S aDVENTERES IÎ WONDEBLAND

  $     $       Lewis Car2oll

   ¢           DHE ÍILDENNIUM`FUMCR▯M EDÉTION 3.0
Š



          !                 CHAPÔE▯  I

       "              Down tèe Rabbht-Holg


▯  Alibe was beginning to get tery tired of sittinG by her
sister
on the bank, and of having nothing to do:  once or twice she
hAd
peepåd into tHe boOk her ñister`was(reating, buu mt had no
pictures oretionq in it, `an` whad is the uce of a book,'
▯thgught ▯lice `wétøout phcpures or Convg2sAtion?'

▯0 So she war considering in!her own mine (as well as she
cgglä,M
bor the hot day"måde her feeì"vEry(sleep9 and s<upid), w`ether
the pleasur% of making a daa3y-chain`wouìd0be wovth"thd trouble
nf getting up and ðicking the daisies. wh%n suddenli a Wléte
Rabbit`wilh p)nk eyes ran clkse$âi he2.▯

  There Was nOthiîg {o0TERy rmmarkable in ôhat; ngr did Álice
thank it s/ VERI much out ofdèe 7a9 to hear the Òabbit say to
atsemf, `Oh dear!  Oh dear!  I 3h`ll fe |ata!'  (when she
thought
it ofer afterwarls, it occqbreD tï her tjat she ought 4o havd

wondered at this,(but at thg time it all!seemed quat
```

(a)

```
             ALICE'S ADVENTURES IN WONDERLAND

                     Lewis Carroll

           THE MILLENNIUM FULCRUM EDITION 3.0



                       CHAPTER I

                  Down the Rabbit-Hole


  Alice was beginning to get very tired of sitting by her sister
on the bank, and of having nothing to do:  once or twice she had
peeped into the book her sister was reading, but it had no
pictures or conversations in it, `and what is the use of a book,'
thought Alice `without pictures or conversation?'

  So she was considering in her own mind (as well as she could,
for the hot day made her feel very sleepy and stupid), whether
```

(b)

Figure 4.3. Simulated transmission of single 4096-octet PSDU in 802.11ad MCS 6 over AWGN channel: (a) received embedded payload with errors, no FEC and (b) received embedded payload with no errors, LDPC(5/8) FEC.

This scenario was then repeated under the same channel conditions utilizing a $R = 5/8$ LDPC code to protect the embedded payload; although the capacity of the embedded channel was significantly reduced, to 5460 bits or 682 ASCII characters, the error performance improved dramatically. As shown in Figure 4.3b, in this second case, the embedded payload was also recovered without error.

4.3.1 Capacity Analysis FEC-Protected Payload

To improve the reliability of our embedded channel, we encoded our embedded payload with the same LDPC codes employed by the underlying channel. As expected, the application of error correction techniques resulted in significant reductions in the amount of embedded payload that could be carried in each PSDU.

The reduction in capacity was not only due to the need to embed FEC parity bits, but also the fact that only complete FEC codewords could be embedded in the PSDU. While the limitation on embedding complete FEC codewords was originally a product of the simulation environment, which was only configured to transmit a single PSDU, justification for such an implementation exists within best practices described for information-hiding techniques in [15]. Specifically, if the encoded embedded data was spread across multiple PSDU, the loss of any one PSDU could corrupt portions of the embedded channel carried on adjacent frames.

For our initial implementation of FEC, we conducted simulated trials to explore the capacity limits of the embedding methodology where the MCS index selected for the underlying communications system had been intentionally decremented. This implementation provided an opportunity to select a higher-rate FEC code for the embedded data than was being utilized on the underlying channel; this was possible due to the fact that we understood that the channel conditions would require a code with less redundancy.

The total size of the FEC-protected embedded payload, L_E, that could be embedded into a given PSDU is determined by

$$L_E = N_{ECW} D_{ECW} \,, \tag{4.4}$$

where N_{ECW} is the number of complete LDPC codewords that can be embedded in each PSDU, and D_{ECW} is the amount of data carried by each codeword. D_{ECW} is based on the FEC selected to protect the embedded payload

$$D_{ECW} = L_{ECW} R_{EC} \ , \tag{4.5}$$

where R_{EC} is the code rate, and L_{ECW} is the overall codeword length. Due to the possibility of selecting unequal error protection for the embedded payload, R_{EC} may not equal the coding rate of the underlying channel. The number of complete codewords in each PSDU could then determined based on the total embedding capacity of each PSDU from (4.3)

$$N_{ECW} = \left\lfloor \frac{C_{PSDU}}{L_{ECW}} \right\rfloor \ . \tag{4.6}$$

By combining (4.4), (4.5), and (4.6), we derive an updated equation for L_E

$$L_E = L_{ECW} R_{EC} \left\lfloor \frac{C_{PSDU}}{L_{ECW}} \right\rfloor , \tag{4.7}$$

which can also be expressed in terms of C_{CW} as

$$L_E = L_{ECW} R_{EC} \left\lfloor \frac{C_{CW} N_{CW}}{L_{ECW}} \right\rfloor \ . \tag{4.8}$$

4.3.2 Interleaving of Embedded Data

Evaluation of the embedded channel with forward error correction led to the development of an alternate embedding technique that aimed to reduce the performance impact to the underlying communications channel without sacrificing the throughput of the embedded channel [5]. In the original implementation, the FEC-protected payload was embedded into the underlying communications channel in such a way that the full n-bit embedding capacity was utilized in the first f codewords, before embedding the remainder r_s bits of data in the codeword in codeword $(f + 1)$ [5]. A representation of this method is shown in Figure 4.4a with the number of fully embedded codewords, f, calculated as [5]

$$f = \left\lfloor \frac{L_E}{n} \right\rfloor , \tag{4.9}$$

where L_E is the total number of embedded payload bits [5].

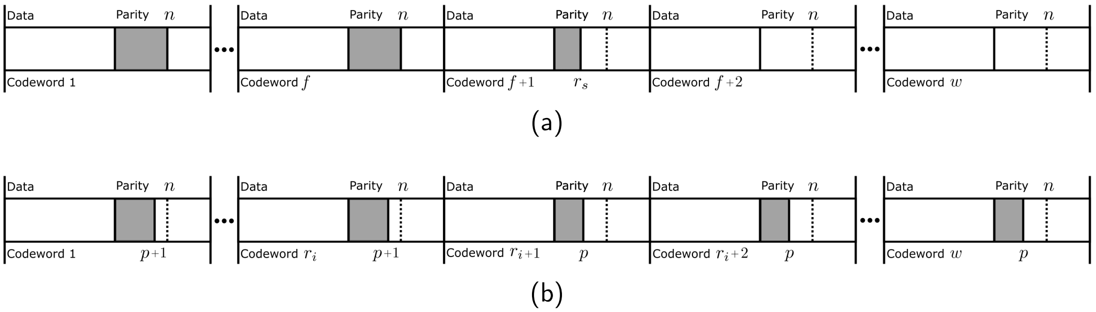


Figure 4.4. Comparison of embedding method for FEC-protected hidden message: (a) standard embedding method and (b) interleaved embedding method. Source: [5], © 2019 IEEE.

The number of bits embedded in the final codeword that contains the payload data, r_s, is determined using the following formula [5]:

$$r_s = L_E - nf = L_E - n\left\lfloor \frac{L_E}{n} \right\rfloor . \tag{4.10}$$

Since the total number of locations available for embedding was greater than the number of bits being embedded, this method resulted in some legitimate codewords having all n parity bits embedded, while others carried 0 embedded bits [5]. This uneven embedding was suboptimal as codewords that were fully embedded are more likely to experience an uncorrectable error, and any codeword errors would ultimately result in a packet error [5].

The new method utilized a process similar to interleaving where the hidden data was distributed equally across all N_{CW} codewords [5]. This method is illustrated in Figure 4.4b where the total number of codewords w, is equal to N_{CW}. Every codeword was embedded with a minimum of p bits [5],

$$p = \left\lfloor \frac{L_E}{w} \right\rfloor , \tag{4.11}$$

where L_E remains the total number of bits being embedded [5]. While all codewords contain at least p-bits, the first r_i codewords will contain $(p + 1)$ bits; r_i is calculated by [5]

$$r_i = L_E - wp = L_E - w\left\lfloor \frac{L_E}{w} \right\rfloor . \tag{4.12}$$

4.4 Improved Estimates of Embedding Capacity

Initial proof of concept trials were conducted to validate the performance of the proposed embedding scheme when operating in the decremented MCS implementation. Since these trials were focused on establishing the performance of our embedding scheme in cases where the selected MCS was intentionally decremented, the focus of the simulations was to determine the maximum level of embedding, measured as bits-per-codeword. The maximum embedding rate was determined by comparing the average PER of the embedded MCS, I_C, at a given channel state to the unembedded performance of the next higher MCS index, I_{C+1}. The embedding rate could be increased so long as the average PER for the embedded MCS did not exceed that of the unembedded MCS for a given SNR. While the data collected from these early trials provided valuable information about the capacity of this specific embedding implementation, it did not provide sufficient granularity to perform mathematical analysis or draw strong conclusions on the behavior of these embedding schemes.

In order to support analysis of our second embedding method, which sought to utilize available M_E within the existing MCS index, it was necessary to significantly expand the number of simulated trials as well as develop a series of analytical techniques to identify the relationship between the channel state, characterized in terms of SNR, and the available embedding capacity. Embedding trials were therefore conducted for all SC DMG MCS at every embedding rate starting at 1-bit-per-codeword up to a maximum rate that was 25% greater than observed during our initial trials; the increased maximum was selected to ensure that we fully explored the embedding range. An example of the results from these embedding trials can be seen in Figure 4.5 where the PER threshold for 802.11ad at the given PSDU-length is indicated by the red-dashed line.

The next step was to determine the SNR required to support each of the embedding rates. Although we could not run a sufficient number of trials to experimentally determine these values, we did note that while the PER manifests as a curve with exponential decay, when viewed on a semi-logarithmic (log-lin) plot, the area of interest around the designated PER threshold appears to be approximately linear. As a result, we were able to utilize a semi-log interpolation technique to estimate the minimum SNR required to achieve a specified PER for a given embedding rate; this interpolation was conducted in MATLAB with the resulting estimated SNR values plotted in Figure 4.6 against the simulated PER

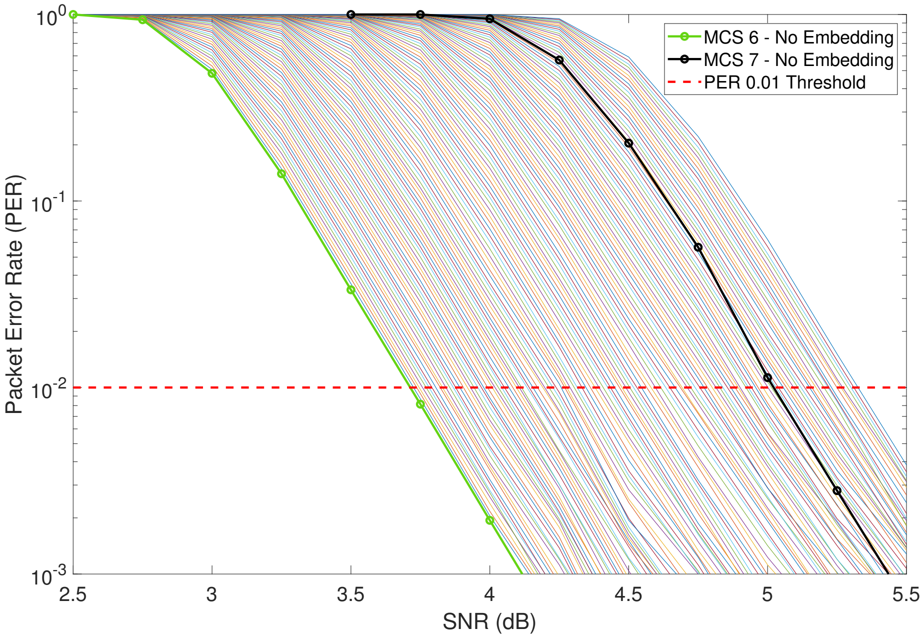


Figure 4.5. Variable rate embedding trials, DMG PHY simulation under AWGN channel. Results for MCS 6, 100000 trials per SNR, 1 to 120 embedded bits per LDPC codeword.

curves. The code utilized to conduct this interpolation can be found in Appendix A.2. The embedding rates obtained from this interpolation, in terms of bits-per-codeword, were then plotted against these SNR estimates in an attempt to quantify the performance of our embedding methodology; an example of the resulting plot is shown in Figure 4.7.

The vertical lines in Figure 4.7 represent the minimum estimated SNR required to maintain the designated PER threshold for MCS 6 and 7. These bounds, which were estimated by performing the same semi-log interpolation technique on the results of simulations conducted against unembedded MCS indices, provide an excellent representation of the performance bounds of our embedding technique for MCS 6. As the number of embedded bits increases, the SNR required to maintain the specified 1% PER also increases. This graphical representation also makes it easy to identify the specific embedding rate where the performance of an embedded channel operating at the current MCS index, I_C, is equivalent to that of the next higher MCS index, I_{C+1}.

Since we are primarily concerned with the embedding capacity between adjacent MCS, we can use the information from Figure 4.7 to identify the upper embedding limit in this trial

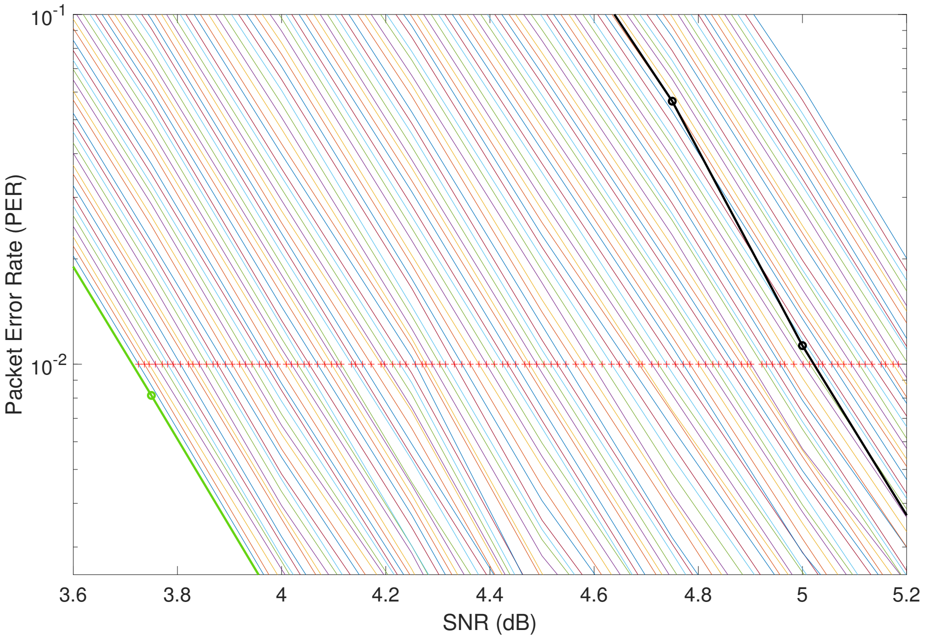


Figure 4.6. Variable rate embedding trials, DMG PHY simulation under AWGN channel. Estimated SNR requirement to achieve 1% PER for each embedding rate. MCS 6, 100000 trials per SNR, 1 to 120 embedded bits per LDPC codeword.

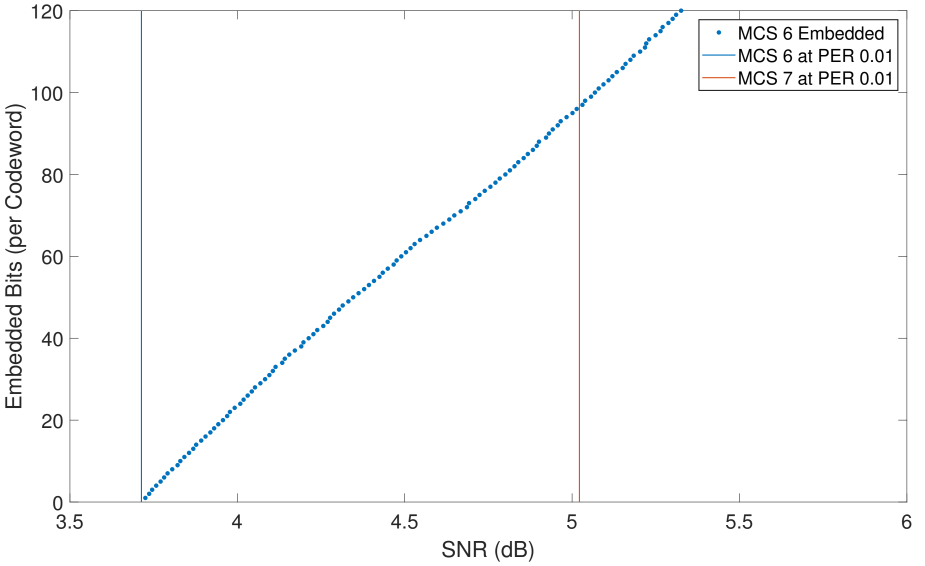


Figure 4.7. Estimated embedding capacity at a given SNR while maintaining 1% PER. DMG PHY simulation under AWGN channel, MCS 6, 100000 PSDU per SNR point, 1 to 120 embedded bits per LDPC codeword.

as 95-bits-per codeword. In an effort to characterize the embedding capacity for each MCS, a linear regression was run against the result of each trial up to this maximum embedding capacity. The results were presented in the standard slope-intercept form

$$y = mx + b, \tag{4.13}$$

where y represents the number of embedded bits per codeword, m is the coefficient that describes the slope of the regression, x represents the channel conditions as described by the SNR, and the constant b provides the y-axis intercept and completes the mathematical description of the line. The resulting line of regression is displayed in Figure 4.8.

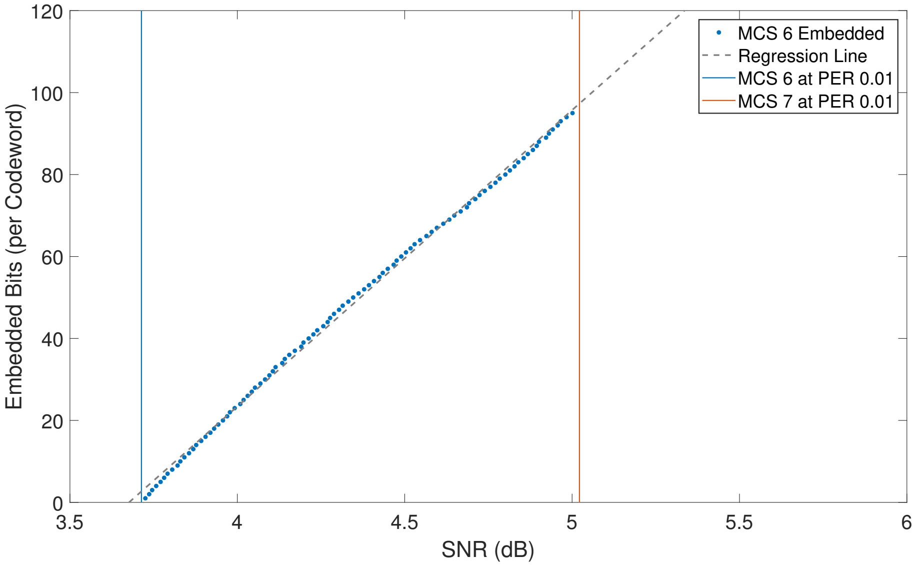


Figure 4.8. Estimated embedding capacity at a given SNR while maintaining 1% PER with associated line of regression. DMG PHY simulation under AWGN channel, MCS 6, 100000 PSDU per SNR point, 1 to 95 embedded bits per LDPC codeword.

When considering the embedding capacity of each MCS, the most significant element of the linear regression is the slope, which represents the number of bits per LDPC codeword that can be embedded for every dB increase in SNR; from this point, we will reference this slope as the estimated embedding coefficient, or $\hat{r}_E$.

Once calculated, the estimated embedding coefficient can be utilized to develop the estimated capacity of each codeword, $\hat{C}_{CW}$, expressed in bits-per-codeword, where

$$\hat{C}_{CW} = \lfloor \hat{r}_E M_E \rfloor, \tag{4.14}$$

and the embedding margin, M_E, is a measure of the difference in SNR between the current channel conditions and the minimum SNR required to maintain the specified PER threshold at the current MCS index for an unembedded PSDU.

Substituting the expression for $\hat{C}_{CW}$ into (4.1) allows us to calculate $\hat{C}_{PSDU}$, an estimated embedding capacity measured in bits-per-PSDU

$$\hat{C}_{PSDU} = N_{CW} \left\lfloor \hat{r}_E M_E \right\rfloor \tag{4.15}$$

which is a function of M_E. Substituting into (4.3) allowed us to obtain an alternate form of the same capacity estimate

$$\hat{C}_{PSDU} = \lfloor \hat{r}_E M_E \rfloor \left\lceil \frac{8\rho L_P}{L_{CW} R_C} \right\rceil . \tag{4.16}$$

While this estimated embedding capacity represents the raw number of bits that can be embedded in a given PSDU, it does not factor in the overhead required to support the error protection for the embedded payload. Finding the length of the estimated FEC-protected embedded payload, $\hat{L}_E$, that can be carried in a single PSDU for a given M_E and $\hat{r}_E$ can be found by substituting the results of (4.14) into (4.8)

$$\hat{L}_E = L_{ECW} R_{EC} \left\lfloor \frac{\lfloor \hat{r}_E M_E \rfloor N_{CW}}{L_{ECW}} \right\rfloor . \tag{4.17}$$

4.5 Embedding Distortion

In addition to the capacity of the embedding payload, another important consideration is the impact these techniques have on the underlying communications channel. Similar to the concept of distortion in traditional steganography, which is a measure of the amount of modification that has been performed on the cover object [23], we will examine the impact of our embedding process in terms of observable changes to the performance of the wireless communication system.

4.5.1 Distortion Types

Our proposed embedding occurs at the physical layer and all traces of our embedded data should be removed from the legitimate information bits before they are passed from the PHY. With this in mind, we have identified three potential impacts that would still be observed even if the embedded data is successfully removed at the receiver.

The first impact, designated as Type 1, is the least significant and occurs when embedding is conducted without decrementing the MCS index. In this case, the measured PER is higher than expected for a given SNR; this distortion, previously identified in Figure 3.8 was labeled as D_E. The second type of distortion, Type 2, is a direct result of the intentional MCS degradation. In this case, the underlying communication system will be operating at a lower throughput than would be expected for the current channel conditions. The final distortion, Type 3, occurs if the embedding impacts the system to such an extent that the PER consistently exceeds the established protocol thresholds. While each vendor implementation of link adaptation methodology is different, if the PER exceeds the prescribed threshold, the communication system will experience excessive retransmissions and lower overall throughput. It is worth noting that the ability to recognize the presence of Type 1 or Type 2 distortion requires access to accurate channel state information or the ability to estimate the current SNR of the received signal.

4.5.2 Distortion Regions

Examining the embedding model outlined in Section 3.4, it is possible to delineate regions that correspond to each type of distortion described above. These regions, identified in Figure 4.9, provide insight into both the amount and types of impact that can be expected on the underlying communication system. If the embedding is being utilized for the purposes of developing a covert channel, the level of distortion can also be an indicator of how vulnerable the channel will be to detection.

The most advantageous region, highlighted in green, is the triangular region where only Type 1 distortion is present. This region offers a lower embedding margin, M_E, than the

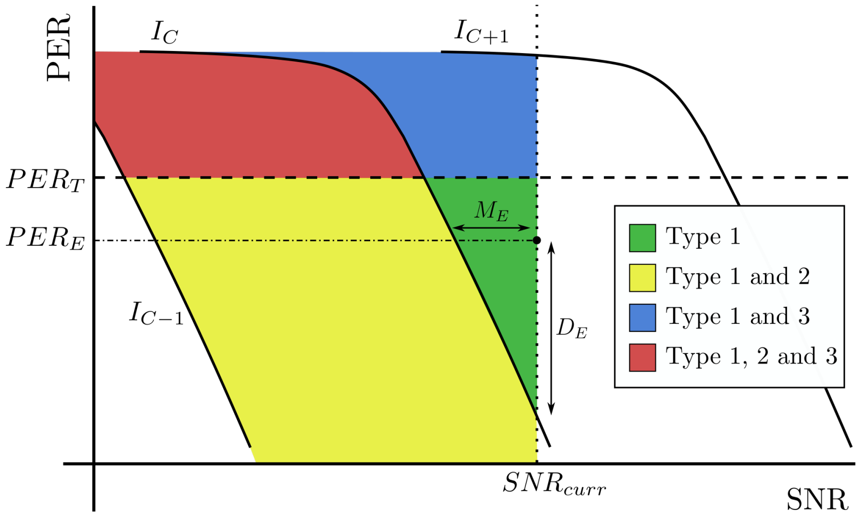


Figure 4.9. Distortion regions for embedding in adaptive rate communication system

case where the MCS is decremented, but results in minimal impact on the channel. In this particular case, the embedding point, identified as the intersection of the embedding PER, PER_E, and SNR_{curr} was selected in a location that reduces the magnitude of distortion, D_E, at the expense of M_E and therefore embedding capacity.

If a higher embedded throughput is required, and the MCS is decremented, the channel will move into the yellow region that contains not only Type 1 distortion but also results in a lower throughput of the underlying channel (Type 2). The final cases, identified as the blue and red regions, occur when the overall PER exceeds the established protocol thresholds. Both of these regions represent the most significant distortion contribution, as the underlying system will suffer significant disruption and loss of throughput.

4.6 Capacity Refinements

This section aims to conduct a closer examination of the constraints associated with selecting an embedding rate within the region of the model that only contains Type 1 distortion. Before being subject to additional constraints, this region is defined by three distinct bounds. The upper horizontal bound, PER_T, represents the protocol threshold for packet error ratio.

The vertical bound along the right-side of the region is defined by the current channel state, SNR_{curr}. Finally, the diagonal bound on the left-hand edge is the performance of the current MCS index, I_C. While these limits adequately describe the region, there are additional constraints that serve to further limit embedding.

It is important to recognize that our ability to move within this region is entirely dependent upon the selection of an embedding rate. Increasing the embedding rate will cause the performance curve of the current MCS index, to move in the direction of the intersection between PER_T and SNR_{curr}. This curve will remain approximately parallel to I_C and will be used to describe the expected PER of the underlying system, quantify the type and magnitude of distortion which results from the embedding, and define M_E which is used to find the embedded channel capacity.

4.6.1 Constraints

An enlarged version of the embedding region subject to Type 1 distortion is shown in Figure 4.10. This triangular region, described by the vertices D, E, and F, is subject to three constraints that serve to reduce the size of the embedding region.

The first constraint is based on the inherent limitations of channel estimation and the impact of this uncertainty on our embedding limits. The IEEE 802.11ad DMG PHY utilizes a preamble composed of a STF and a CEF; these fields employ Golay sequences to perform synchronization, automatic gain control and channel estimation in the time and frequency domain [46]. While the use of Golay sequences provides robust channel estimation, as noted in Chapter 2, there are also SNR estimation considerations related to the specific channel environment [39] and equipment calibration [42]. As a result, when evaluating the embedding region we propose establishing an offset, ϵ, from the estimated SNR_{curr}. While this offset will reduce both M_E and the maximum embedding capacity, it will also reduce the chance of unintentionally introducing uncorrectable errors. Revisiting (4.14), the estimated embedding capacity per codeword due to this offset, $\hat{C}_{CW_\epsilon}$, can be represented as

$$\hat{C}_{CW_\epsilon} = \lfloor \hat{r}_E \left(M_E - \epsilon \right) \rfloor \ . \tag{4.18}$$

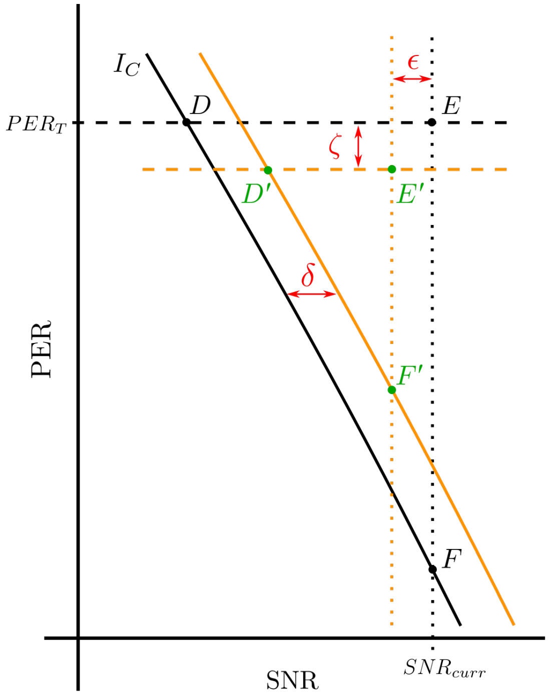


Figure 4.10. Visualization of practical embedding region subject to additional constraints

The second constraint is a user-defined factor of safety that imposes a more restrictive packet error threshold than that specified by the protocol standard, PER_T. Lowering the upper acceptable PER limit by ζ reduces the embedding capacity of the channel but also reduces the likelihood that embedding will cause the underlying system to exceed the established PER limits.

The PER performance of the underlying channel in the vicinity of the specified PER_T has been shown to be approximately linear on a semi-logarithmic plot; as long as the updated $PER_{T-\zeta}$ remains within this log-linear region, the estimated embedding coefficient will remain relatively consistent. This behavior can be observed in Figure 4.11 where embedding results are plotted for thresholds of PER = 0.01, PER = 0.005, and PER = 0.0025. While the minimum required SNR increases as the PER threshold is decreased, the slope of the resulting $\hat{r}_E$ remains relatively unchanged. The $\hat{r}_E$ derived from the example in Figure 4.11 are shown below in Table 4.2.

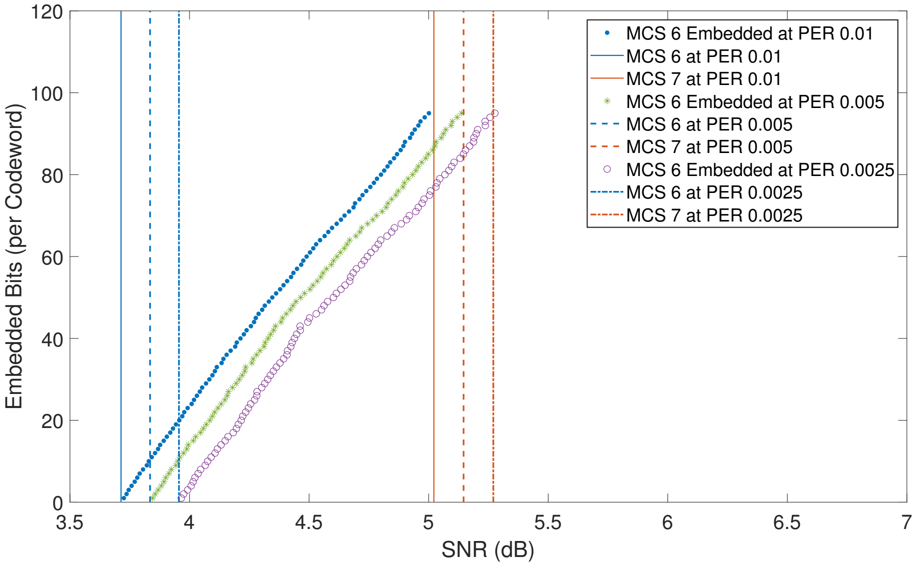


Figure 4.11. Comparison of embedding rates for DMG PHY simulation under AWGN channel at various packet error ratio thresholds. MCS 6, 100000 PSDU per SNR, 1 to 95 embedded bits per LDPC codeword.

Table 4.2. Estimated embedding coefficient for IEEE 802.11ad DMG MCS 6 in AWGN at varying PER thresholds

PER Threshold	**Estimated Coefficient of Embedding, $\hat{r}_E$**
0.01	72.33
0.005	71.49
0.0025	70.59

Assuming a consistent value for $\hat{r}_E$, we can update (4.18) to reflect the reduced margin of embedding, M_{E_ζ}, at the new PER threshold, $PER_{T-\zeta}$

$$\hat{C}_{CW_{max}} = \left\lfloor \hat{r}_E \left(M_{E_\zeta} - \epsilon \right) \right\rfloor , \tag{4.19}$$

and provide an estimated maximum capacity per codeword, $\hat{C}_{CW_{max}}$, which accounts for the offsets of ζ and ϵ.

The final constraint on the embedding region, δ, represents the minimum embedding margin, $M_{E,min}$, required to support the required throughput for the embedded channel,

U_E, measured in bits-per-second. To develop our estimate for δ, we need to make the following assumptions about our visibility into the MAC-layer to include the number of PSDU, N_{PSDU}, transmitted to the destination STA during each DMG beacon interval (BI), and the duration of the BI, T_{BI}, in seconds. Given this information, we can determine minimum FEC-protected payload, $L_{E,min}$, that must be carried by each PSDU:

$$L_{E,min} = \left\lceil \frac{U_E T_{BI}}{N_{PSDU}} \right\rceil . \tag{4.20}$$

We must next determine the minimum capacity, $C_{PSDU,min}$, measured in bits-per-PSDU, required to successfully embed our payload

$$C_{PSDU,min} = L_{ECW} \left\lceil \frac{L_{E,min}}{L_{ECW} R_{ECW}} \right\rceil , \tag{4.21}$$

using a R_{ECW}-rate LDPC code with L_{ECW}-bit codewords. It is important to note that this equation ensures that the minimum embedding capacity is sufficient to embed at least one FEC codeword independent of the required U_E.

Using insight from (4.1), we can determine the minimum embedding capacity, $C_{CW,min}$, measured in bits per codeword

$$C_{CW,min} = \left\lceil \frac{C_{PSDU,min}}{N_{CW}} \right\rceil , \tag{4.22}$$

given the number LDPC codewords in each PSDU, N_{CW}, which is a function of both the selected MCS index as well as the PSDU length, L_P. Using (4.14), we established the relationship of (4.22) with $M_{E,min}$

$$C_{CW,min} = \left\lfloor \hat{r}_E M_{E,min} \right\rfloor \tag{4.23}$$

and the estimated embedding coefficient $\hat{r}_E$. Since $C_{CW,min}$ was an integer value we can transform the equation

$$C_{CW,min} \leq \hat{r}_E M_{E,min} < C_{CW,min} + 1\,, \tag{4.24}$$

which can be further simplified given that $\hat{r}_E > 0$ as

$$\frac{C_{CW,min}}{\hat{r}_E} \leq M_{E,min} < \frac{C_{CW,min} + 1}{\hat{r}_E}\,, \tag{4.25}$$

where $M_{E,min}$ could be any value on the interval

$$\left[\frac{C_{CW,min}}{\hat{r}_E}, \frac{C_{CW,min} + 1}{\hat{r}_E}\right)\,. \tag{4.26}$$

Considering the intent of the offset, δ, as it relates to the concept of $M_{E,min}$, we selected the upper bound to ensure we maintained sufficient capacity to support U_E; therefore,

$$\delta = M_{E,min} = \frac{C_{CW,min} + 1}{\hat{r}_E}\,, \tag{4.27}$$

or

$$\delta = \frac{1}{\hat{r}_E}\left(C_{CW,min} + 1\right)\,. \tag{4.28}$$

We then sought to derive an equation for δ as an expression of U_E, as well as attributes of the current MCS index and the FEC code selected to protect the embedded payload. We first substituted values from (4.22) for $C_{CW,min}$ to obtain

$$\delta = \frac{1}{\hat{r}_E}\left\lceil\frac{C_{PSDU,min}}{N_{CW}}\right\rceil + \frac{1}{\hat{r}_E}\,, \tag{4.29}$$

replacing $C_{PSDU,min}$ with the expression from (4.21)

$$\delta = \frac{1}{\hat{r}_E}\left\lceil \frac{L_{ECW}}{N_{CW}} \left\lceil \frac{L_{E,min}}{L_{ECW}R_{ECW}} \right\rceil \right\rceil + \frac{1}{\hat{r}_E}, \tag{4.30}$$

before replacing $L_{E,min}$ from (4.20)

$$\delta = \frac{1}{\hat{r}_E}\left\lceil \frac{L_{ECW}}{N_{CW}} \left\lceil \frac{\left\lceil \frac{U_E T_{BI}}{N_{PSDU}} \right\rceil}{L_{ECW}R_{ECW}} \right\rceil \right\rceil + \frac{1}{\hat{r}_E}. \tag{4.31}$$

Since the value of $L_{ECW}R_{ECW}$, which represents the amount of data bits carried in each LDPC codeword, is an integer, and the value of $U_E T_{BI}$ and N_{PSDU} are real numbers [74],

$$\delta = \frac{1}{\hat{r}_E}\left\lceil \frac{L_{ECW}}{N_{CW}} \left\lceil \frac{U_E T_{BI}}{N_{PSDU}L_{ECW}R_{ECW}} \right\rceil \right\rceil + \frac{1}{\hat{r}_E}. \tag{4.32}$$

Finally, substituting the expression for N_{CW} from (4.2) yields

$$\delta = \frac{1}{\hat{r}_E}\left\lceil \frac{L_{ECW}}{\left\lceil \frac{8\rho L_P}{L_{CW}R_C} \right\rceil} \left\lceil \frac{U_E T_{BI}}{N_{PSDU}L_{ECW}R_{ECW}} \right\rceil \right\rceil + \frac{1}{\hat{r}_E}. \tag{4.33}$$

4.6.2 Edge Cases

The vertical boundary between vertices E' and F' on the edge of the reduced region in Figure 4.10 represents an estimate of the current channel state and the location that will maximize M_E for a given PER. The horizontal edge of the region, connecting D' and E', identifies the maximum permissible PER based on both the protocol standard and any specified factor of safety. Finally, the diagonal segment connecting D' and F' is the expected performance of the communications channel while supporting the minimum embedding rate required to meet the desired embedded channel throughput.

The vertices of this reduced region that intersect the adjusted SNR limit represent locations of either minimum distortion or maximum embedding capacity. Specifically, F' represents the lowest Type 1 distortion while supporting the lowest allowable embedding capacity. Conversely, E' represents the maximum possible M_E, and therefore the highest capacity, but achieves this performance at the highest level of distortion.

4.7 Implementation of Multipath Fading Channel

The final modification to our 802.11ad testbed involved the implementation of a multipath fading channel to better approximate the performance of our embedding methodologies under realistic channel conditions. The channel model accepted by IEEE P802.11 Task Group ad (TGad) for the DMG protocol is outlined in [75]; unfortunately, while the MATLAB WLAN Toolbox contains system objects to model fading channels for many modern 802.11 standards, they did not develop a specific implementation for 802.11ad DMG.

Fortunately, the 2018b and 2019a releases of the WLAN Toolbox contained a channel model for IEEE 802.11ay based on [76], which was developed to support IEEE P802.11 TGay. Although this model supports characteristics that are not implemented by SC DMG, to include MIMO, both [76] and the MATLAB implementation define a legacy single-input and single-output (SISO) mode [77]. Furthermore, the updated model included confirmation that both the polarization and space-time characteristics of the propagation channel had been correctly implemented in the original TGad model [76].

The MATLAB implementation of [76] did not incorporate the three existing 802.11ad channel models *Living Room*, *Enterprise Cubicle*, or *Conference Room* that remained in the 802.11ay revision; instead they opted to implement three of the newer models including *Open Space*, *Street Canyon*, and *Large Indoor Space (Hotel Lobby)* [77]. To best align with the characteristics of SC DMG, we selected the open area hotspot implementation, which contains only a single reflection plane, and the single user SISO (SU-SISO) user characteristic. Based on expected range estimates from [47], we selected the locations of our stations to achieve a slant range of approximately 10 meters. In this configuration, the MATLAB implementation conducts ray tracing to develop the LOS and non-line-of-sight (NLOS) paths with a single first-order reflection. The simulated results that employed this model are presented in Section 6.3.1

4.8 Summary

In this chapter, we introduced our proposed implementation of error correction code-based embedding within 802.11ad DMG and identified the critical aspects of developing a MATLAB-based simulation to ascertain the performance of our techniques. We then discussed the use of error correction codes to improve the bit-error performance of our embedded payload and derived expressions to determine the maximum FEC-protected payload capacity. Analysis techniques were developed to estimate the embedding capacity based on the channel state and then expanded to consider constraints that might limit the size of the embedded payload as well as identify the expected distortion on the underlying communication system. Finally, we outlined the implementation of a simulated multipath fading channel to establish the performance of our embedding techniques under more realistic channel conditions. Relevant results obtained from the simulations developed for these LDPC-based embedding techniques will be presented in Chapter 6.

CHAPTER 5: Error Correction-Based Embedding in Convolutional Codes

Following our investigation of LDPC codes, we turned our attention to developing embedding techniques within convolutional codes. We specifically chose to examine the $R = 1/2$ (7, [133, 171]) convolutional code, first proposed in [78], that has a constraint length $K = 7$, and an encoder that can be described by the polynomial generators $g_0 = 133$ and $g_1 = 171$. This convolutional code has seen broad application in commercial communications systems and has been utilized within Wi-Fi standards as far back as IEEE 802.11a. Unlike the LDPC codes utilized for DMG, this BCC is not systematic and therefore, there is no distinction between data and parity bits in the output of the encoder.

The use of this specific error correction code within IEEE 802.11 is heavily dependent on puncturing techniques to facilitate increased data rates. In the most recent standards, this code can be found with three distinct puncture rates of $R = 2/3$, $R = 3/4$, and $R = 5/6$. Consequently, any embedding techniques developed for this particular code had to be capable of being implemented against both the unpunctured $R = 1/2$ parent code as well any of the punctured rates.

Our work on convolutional codes afforded us the opportunity to develop a simplified model of an adaptive rate communication system. The attributes of our model were carefully selected to maximize utility while minimizing complexity. Once we validated our embedding concepts within this model, we applied portions of these techniques in a MATLAB testbed for IEEE 802.11ac VHT.

Despite the fact that convolutional codes function in an entirely different manner than block codes, we retained the three key concepts from our original embedding methodology. First, our embedding technique would seek to remove our embedded data at the receiver prior to decoding. Second, we would explore implementations that would require the selection of sub-optimal MCS indices to increase the embedding capacity. Third, we would attempt to develop techniques that could exploit available capacity within the current MCS index. A block diagram of the information-hiding architecture within a generic convolutional error correction code-based communication system is shown in Figure 5.1.

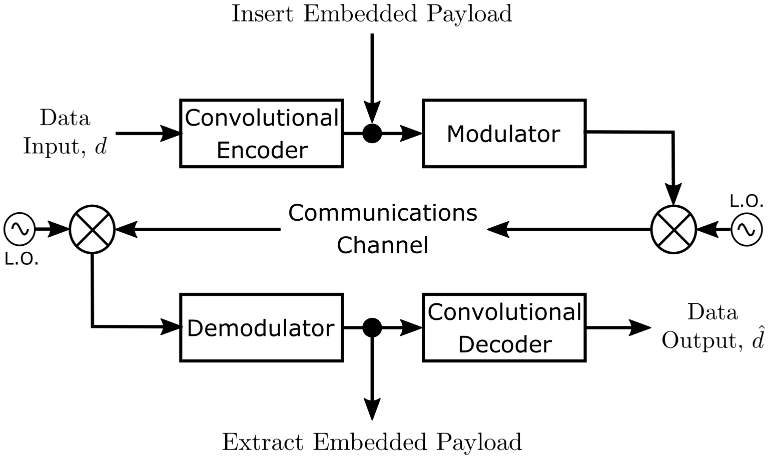


Figure 5.1. Major components of generalized convolutional code-based embedding process

5.1 Adaptive Rate Embedding Model

During the development and testing of the complex simulation which was utilized to implement our embedding techniques against IEEE 802.11ad, we made a number of observations related to the behavior of both the underlying communications channel and the embedded payload that pertain to data collection and analysis.

We observed that the BER for an uncoded embedded payload was consistent with the performance of uncoded data transmitted at the specified modulation type and channel state. If FEC was implemented to protect the embedded payload, the performance of that error correction mechanism was simply a factor of the selected code rate and the current channel conditions. Most importantly, the impact of embedding on the performance of the underlying communication channel, measured in PER, was solely determined by the number and location of the embedded bits; specifically, the critical factor was the impact of the erasures in the received code that occurred when we extracted our embedded payload from the underlying channel. The actual content of those bit locations during the simulated transmission and demodulation did not have any impact on PER.

These observations were used to make deliberate design decisions with respect to the development and implementation of our embedding techniques; specifically, for our initial trials, no data would actually be embedded in the transmitted frames. So long as we removed the bit values prior to decoding, the performance impact to the underlying system would be accurate but the complexity of the testbed would be significantly reduced.

An additional consideration in the development of our model was the selection of the modulation type and channel model. By selecting binary phase-shift keying (BPSK) modulation over an AWGN channel, we were able to leverage well understood analytical expressions to derive the error bounds of our specified $(7, [133, 171])$ convolutional code. Not only would these equations allow us to validate the implementation of our adaptive rate model, and compare estimated BER of our notional MCS indices against the analytically derived performance bounds, but also enable us to calculate the expected error performance of our embedded payload without relying on experimental results.

As a result, our proposed adaptive rate system would have four distinct rates, designated MCS A, MCS B, MCS C and MCS D; the MCS indices would each adopt a coding rate and puncture pattern that aligned with the real world implementation of the $(7, [133, 171])$ convolutional code. MCS A would utilize the parent code with $R = 1/2$, with MCS B, C and D utilizing rates $R = 2/3$, $R = 3/4$, and $R = 5/6$, respectively.

With a notional adaptive rate model established, we leveraged analytical expressions from [59], [63], [65] to establish the theoretical performance bounds. The necessary characteristics to define the performance of $(7, [133, 171])$ are contained in Table 5.1 where d_f is the minimum free distance of the code, a_d is the event weight for the code, and c_d is the information error weight of the code.

Table 5.1. Code description and weight spectra of IEEE 802.11 binary convolutional code. Adapted from [65].

Constraint Length K	**Generators (in octal)**	**Free Distance** d_f	**Weight Spectra** $(a_d, d = d_f, d_f + 1, \ldots, d_f + 19)$ $[c_d, d = d_f, d_f + 1, \ldots, d_f + 19]$
7	133, 171	10	(11, 0, 38, 0, 193, 0, 133, 0, 7275, 0, 40406, 0, 234969, 0, 1337714, 0, 7594819, 0, 43375588, 0) [36, 0, 211, 0, 1404, 0, 1163, 0, 77433, 0, 502690, 0, 3322763, 0, 21292910, 0, 134365911, 0, 843425871, 0]

The event weight, a_d, represents the number of incorrect paths with a distance, d, that diverge from the correct path before rejoining at a later stage where $d \geq d_{free}$; the information error weight, c_d, is a related metric that counts the total number of information bit errors on the paths represented by a_d.

It is then possible to calculate the probability of bit error, P_b, union bound [59]

$$P_b < \sum_{d=d_{free}}^{\infty} c_d P_d \tag{5.1}$$

where P_d is the probability of selecting an error path of weight d. Although the summation term in (5.1) is infinite, the first terms of the series are dominant and extremely accurate estimates for P_b can be calculated after as few as ten terms [79]. The expression for P_d is dependent upon the modulation type and the channel itself. In the simplest case, P_d for hard decision decoding of a binary symmetric channel is given by [63]

$$P_d = \left[2\sqrt{p\,(1-p)}\right]^d \tag{5.2}$$

where p is the probability of channel transition. A more appropriate estimate for our model can be found by utilizing the pairwise error probability for coherent soft-decision BPSK in AWGN [65]

$$P_d = Q\left(\sqrt{2d\frac{E_b R}{N_0}}\right), \tag{5.3}$$

where d remains the weight of the error path, E_b is the energy per information bit, N_0 is the noise spectral density, and R is the code rate. Combining (5.1) and (5.3) yields

$$P_b < \sum_{d=d_{free}}^{\infty} c_d Q\left(\sqrt{2d\frac{E_b R}{N_0}}\right). \tag{5.4}$$

An improved bound for the performance of punctured convolutional codes relies upon the fact that the puncture period of the code imparts a regular time-variance based on the

Table 5.2. Code description and weight spectra of punctured rates of IEEE 802.11 binary convolutional code. Adapted from [80].

Puncturing Rate R	**Puncturing Pattern**	**Free Distance** d_f	**Weight Spectra** $(a_d, d = d_f, d_f + 1, \ldots, d_f + 9)$ $[c_d, d = d_f, d_f + 1, \ldots, d_f + 9]$
2/3	$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$	6	(1, 16, 48, 158, 642, 2435, 91274, 34705, 131585, 499608) [3, 70, 285, 1276, 6160, 27128, 117019, 498860, 2103891, 8784123]
3/4	$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$	5	(8, 31, 160, 892, 4512, 23307, 121077, 625059, 3234886, 16753077) [42, 201, 1492, 10469, 62935, 379644, 2253373, 13073811, 75152755, 428005673]
5/6	$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$	4	(14, 69, 654, 4996, 39699, 315371, 2507890, 19921920, 158275483, 1257455600) [92, 528, 8694, 79453, 792114, 7375573, 67884974, 610875423, 5427275376, 47664215639]

starting position within the puncture period k [65]. This revised expression for the union bound [65]

$$P_b < \frac{1}{k} \sum_{d=d_{free}}^{\infty} c_d P_d \,, \tag{5.5}$$

was then used along with the distance spectra in Table 5.2 to calculate the performance boundaries of the codes in our adaptive rate model. The resulting bounds for P_b are shown in Figure 5.2.

Once we had an analytical description of the performance bounds of the convolutional codes, we developed a simple MATLAB script to calculate BER performance for each of these code rates under BPSK modulation over an AWGN channel. The software testbed, which can be found in Appendix A.3, was configured to employ soft decision demodulation to generate approximate LLR values that were then passed through a Viterbi decoder. A

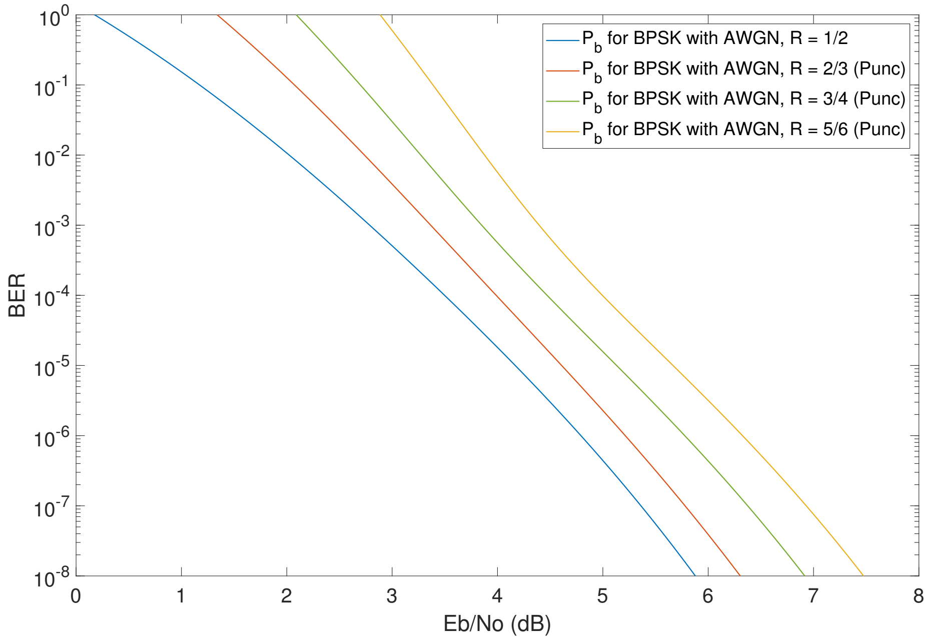


Figure 5.2. Probability of bit error upper union bound for BSPK over AWGN with $R = 1/2$ convolutional code $(K = 7,\ g_0 = 133,\ g_1 = 171)$ with punctured rates of $R = 2/3$, $R = 3/4$, and $R = 5/6$

traceback depth of $\tau = 96$ was utilized for all code rates.

Simulations were then conducted for each of our notional MCS indices over a range of E_b/N_0 values under AWGN conditions. We selected a frame size of 1200 bits, which contained randomly generated binary data, and computed the average BER at the testbed receiver. The number of frames transmitted for each E_b/N_0 value varied depending on the observed error performance. For each trial, data frames would be encoded and transmitted over the channel at each E_b/N_0 point until a total of 1000 bit errors were observed; if this threshold was not reached, the testbed would continue transmitting frames until the total number of transmitted bits exceeded 1×10^8. This implementation was adapted from similar examples published by MATLAB and allowed us to reduce the overall time required for the simulation while obtaining high fidelity results. These trials were then completed multiple times under different channel conditions. The BER curves resulting from these trials are shown in Figure 5.3 when plotted against the error bounds for convolutionally coded BPSK from (5.5).

The BER derived from the simulated trials of our notional MCS indices clearly converge

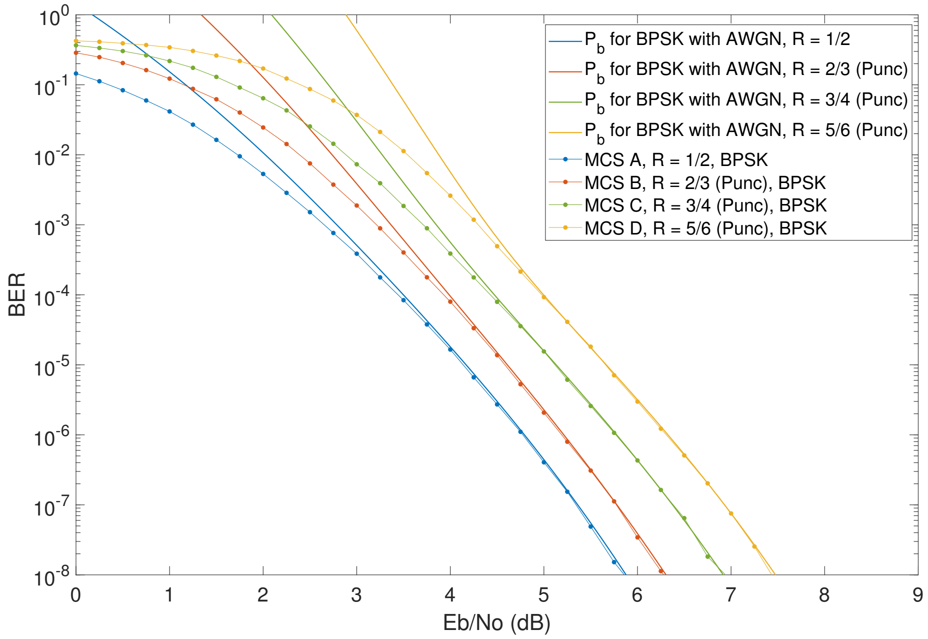


Figure 5.3. Performance of adaptive rate model MCS indices compared to theoretical limits, BER versus E_b/N_0, 100 trials

.

on the upper union bound for P_b; this result validated the performance of our MATLAB BPSK testbed and facilitated further testing of the proposed embedding implementations.

5.2 Convolutional Code Embedding

Having established the baseline performance of our MCS model, we turned our attention to developing a puncturing implementation for the specified BCC. Separate methods had to be developed depending on whether the communication system was operating at an MCS index that utilized the unpunctured parent code or one that used punctured codes.

With analytical expressions to quantify P_b for the convolutional codes under consideration, we focused our simulation on the performance of the underlying communication channel based on the raw number of embedded bits. As previously discussed, the embedding payload for our MCS model could be simulated by the action of simply removing specific received bits from consideration at the convolutional decoder. In the MATLAB implementation of the Viterbi decoder, this action could be accomplished by either replacing those received bit locations with 0 prior to decoding, or declaring those bit locations as being invalid within the Viterbi decoder structure. Either method ensured the content of those bit locations were not considered during branch metric calculations.

As with our 802.11ad implementation, we sought to exploit excess redundancy provided by the $R = 1/2$, $(7, [133, 171])$ convolutional code; this excess redundancy could exist due to the fact that the current channel conditions simply exceed the minimum requirement for the current MCS, or as the result of the MCS index being intentionally decremented. While the specific implementation was not addressed for the purposes of our adaptive rate model, we did consider embedding implementations to be viable so long as the E_b/N_0 requirements of the embedded MCS, I_C, did not exceed the minimum E_b/N_0 requirements for the next higher MCS, I_{C+1}.

5.2.1 Embedding Within Unpunctured Code

In developing an embedding and extraction method for this BCC, the more trivial case was for the unpunctured parent code. Extensive research has been conducted into optimal puncturing patterns for convolutional codes that maximize both throughput and error performance. Many of these puncture patterns have been developed our $R = 1/2$, $(7, [133, 171])$ code, and any of these could be potentially implemented as an embedding scheme; instead of puncturing bits prior to transmission, the punctured locations would be used to carry our embedded payload. Our embedding scheme would then ensure these embedded bits were removed from consideration prior to the decoding process.

An example of this embedding technique is illustrated in Figure 5.4. In our notional example, nine bits of source data are first passed through the $R = 1/2$ BCC resulting in the output of 18 encoded bits. Our six payload bits are then embedded into the encoded data at the positions outlined in red; these positions are the same locations that would normally be punctured for the $R = 3/4$ code as specified in [44]. This modified data stream is then transmitted across a channel before being passed through the demodulation and decoding process at the receiver. In this particular case, the proposed method would embed six bits for every 18 bits in the coded bit stream, and therefore be expected to deliver an embedded data rate equivalent to one-third of the data rate of the underlying channel.

Given the well-defined nature of the puncturing patterns for the specified parent code, and the fact that analytical expressions exist to characterize code performance, we did

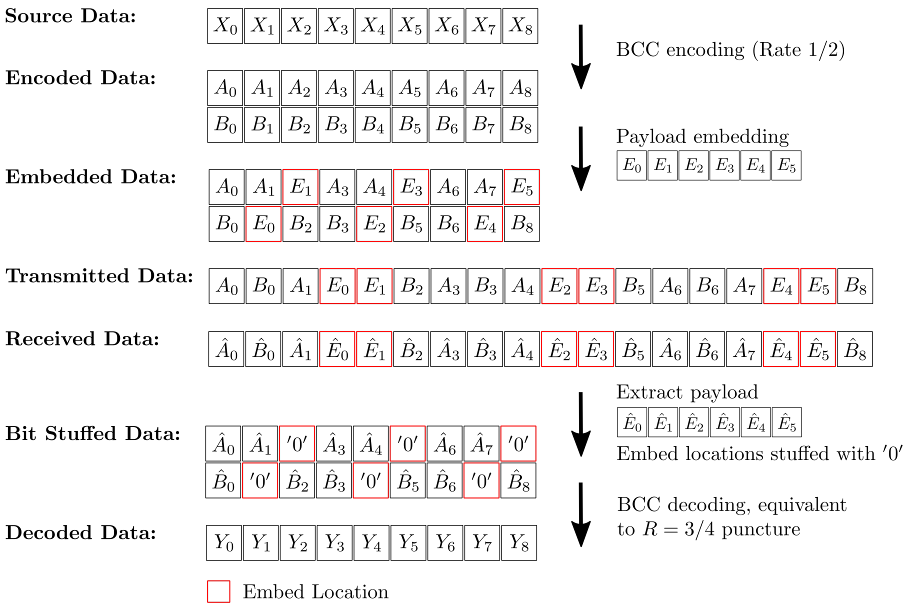

Source Data:
X_0 X_1 X_2 X_3 X_4 X_5 X_6 X_7 X_8
BCC encoding (Rate 1/2)
Encoded Data:
A_0 A_1 A_2 A_3 A_4 A_5 A_6 A_7 A_8
B_0 B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8
Payload embedding
E_0 E_1 E_2 E_3 E_4 E_5
Embedded Data:
A_0 A_1 E_1 A_3 A_4 E_3 A_6 A_7 E_5
B_0 E_0 B_2 B_3 E_2 B_5 B_6 E_4 B_8
Transmitted Data:
A_0 B_0 A_1 E_0 E_1 B_2 A_3 B_3 A_4 E_2 E_3 B_5 A_6 B_6 A_7 E_4 E_5 B_8
Received Data:
$\hat{A}_0$ $\hat{B}_0$ $\hat{A}_1$ $\hat{E}_0$ $\hat{E}_1$ $\hat{B}_2$ $\hat{A}_3$ $\hat{B}_3$ $\hat{A}_4$ $\hat{E}_2$ $\hat{E}_3$ $\hat{B}_5$ $\hat{A}_6$ $\hat{B}_6$ $\hat{A}_7$ $\hat{E}_4$ $\hat{E}_5$ $\hat{B}_8$
Extract payload
$\hat{E}_0$ $\hat{E}_1$ $\hat{E}_2$ $\hat{E}_3$ $\hat{E}_4$ $\hat{E}_5$
Embed locations stuffed with '0'
Bit Stuffed Data:
$\hat{A}_0$ $\hat{A}_1$ '0' $\hat{A}_3$ $\hat{A}_4$ '0' $\hat{A}_6$ $\hat{A}_7$ '0'
$\hat{B}_0$ '0' $\hat{B}_2$ $\hat{B}_3$ '0' $\hat{B}_5$ $\hat{B}_6$ '0' $\hat{B}_8$
BCC decoding, equivalent to $R = 3/4$ puncture
Decoded Data:
Y_0 Y_1 Y_2 Y_3 Y_4 Y_5 Y_6 Y_7 Y_8
□ Embed Location

Figure 5.4. Proposed embedding scheme for unpunctured convolutional codes. Embedding conducted on $R = 1/2$ code resulting in an equivalent $R = 3/4$ rate code.

not conduct exhaustive simulations for this embedding technique. Instead, we focused on quantifying the capacity of this embedding method in terms of the equivalent puncture rate selected for embedding.

5.2.2 Embedding within Punctured MCS Indices

For the punctured code rates, an alternative method was needed to deliver the optimal embedding rate while selecting embedding locations that still allow the code to deliver predictable error performance to the underlying communication system. We recognized that since we would be conducting our embedding in the bits that remain after the puncturing process, our embedding technique had to factor in the location of the punctured bits.

Although not identical in application, we drew inspiration for our embedding technique from the ability of RCPC to support unequal embedding rates while maintaining the same encoder and decoder structure. Recall that in RCPC, lower rate codes are generated from higher rate codes by adding back additional bits that had been previously punctured. We investigated the possibility of conducting the inverse process on the $R = 1/2$, $R = 3/4$ and

$R = 5/6$ punctured codes in our adaptive rate model, but all of the punctured code rates in our model were already reduced to a structure, $R = k/n$ where $n - 1 = k$. Consequently, there were not any viable locations that could be embedded in every puncture period, k, without removing all redundancy from the code. Instead, we proposed appending the output from multiple puncture periods into a block, and then selecting a single embedding location from the resulting code sequence. This method would greatly reduce the impact on the code redundancy at the expense of our embedding capacity.

We then proceeded to assess two of the punctured codes rates, $R = 2/3$ and $R = 3/4$, to determine the optimal block size and embedding location. As a starting point, we recognized that since the impact of our proposed embedding mechanism mimics the behavior of punctured bit locations at the decoder, we could develop an expression to describe our proposed embedding scheme as an equivalent punctured rate. By appending m puncture periods, and selecting a single embedding location, the equivalent coding rate, R_{equiv}, was found to be

$$R_{equiv} = \frac{mk}{mn - 1} \tag{5.6}$$

where n is the number of coded data bits in each puncture period of the base coding rate. We looked at developing an equivalent puncture pattern which would provide BER performance that fell between the performance curves in our current model and allowed us to take advantage of an implementation where we were operating at a decremented MCS, or situations where the channel conditions exceeded the minimum requirements for the current MCS. Possible equivalent puncture rates were examined for various values of m; candidate rates for the base codes of $R = 2/3$ and $R = 3/4$, determined from (5.6), are listed in Table 5.3.

The $R_{equiv} = 10/14$ and $R_{equiv} = 12/15$ were selected for implementation; their selection was also influenced by the fact that previous work in [65], [80] had published the weight spectra for puncturing patterns for our BCC at rates equal to the reduced fraction form of our proposed R_{equiv}. These analytically derived P_b bounds for the puncture rates of $R = 5/7$ and $R = 4/5$ would then be available to help assess the performance of the proposed embedding pattern.

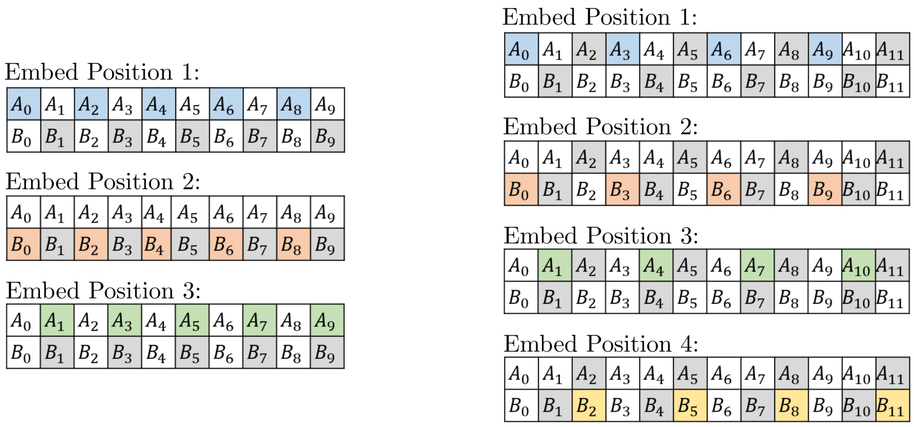
Table 5.3. Equivalent puncture rate, R_{equiv}, achieved for appending m-puncture periods for base code rates of $R = 2/3$ and $R = 3/4$; single embedded bit per block.

m	**Base Code Rate**	
	$R = 2/3$	$R = 3/4$
1	$R_{equiv} = 2/2$	$R_{equiv} = 3/3$
2	$R_{equiv} = 4/5$	$R_{equiv} = 6/7$
3	$R_{equiv} = 6/8$	$R_{equiv} = 9/11$
4	$R_{equiv} = 8/11$	$R_{equiv} = 12/15$
5	$R_{equiv} = 10/14$	$R_{equiv} = 15/19$
6	$R_{equiv} = 12/17$	$R_{equiv} = 18/22$q

To verify the performance of the proposed embedding scheme, we utilized our MATLAB-based MCS model testbed to conduct a series of simulations across an AWGN channel; any non-punctured bits in the series of m-concatenated puncture periods of the base code was eligible for embedding. All of the available embedding locations, a total of 15 in $R_{equiv} = 10/14$, and 16 in $R_{equiv} = 12/15$, were tested within the MATLAB model. The resulting performance curves were then utilized to identify the optimal embedding location. The trials were structured to first compare each of the groups of embedding positions as shown in Figure 5.5; each group was identified by their relative location within the output bitstream of the original code. The punctured locations from the parent code, which are not available for embedding, are highlighted in gray.

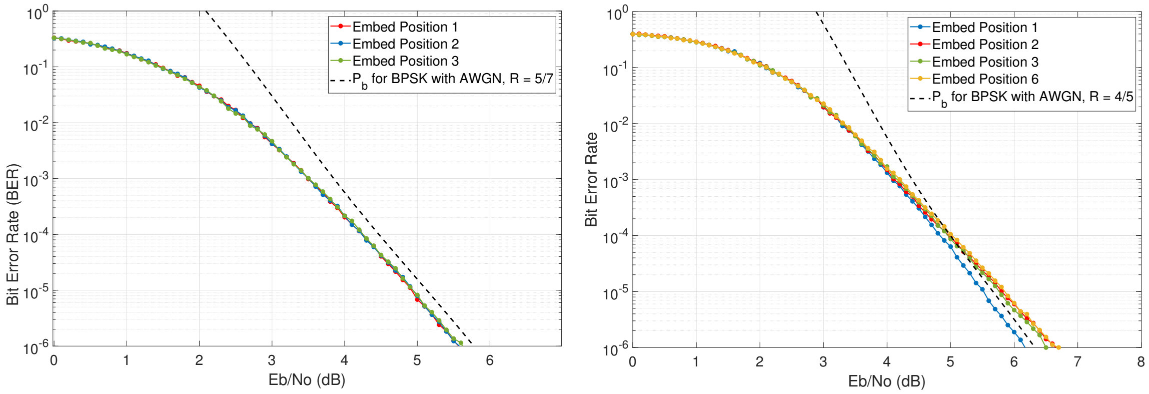
Trials conducted utilizing our MCS testbed did not find any meaningful differences between locations selected within the same embedding group. When the embedding groups were compared against one another, as shown in Figure 5.6, the only significant outlier was the first embedding location for $R_{equiv} = 12/15$, which appeared to deliver a lower BER at increased E_b/N_0. We were also able to compare these embedding results against the P_b bounds for comparable rate punctured codes. We compared $R_{equiv} = 10/14$ against a $R = 5/7$ code from [80] and $R_{equiv} = 12/15$ against a $R = 4/5$ code from [65] and found that both embedding locations compared favorably to the analytically derived P_b bounds.

Ultimately, we selected the second embedding location in the first puncture period for both implementations. Although the first embedding location for the $R_{equiv} = 12/15$


Embed Position 1:
A_0 A_1 A_2 A_3 A_4 A_5 A_6 A_7 A_8 A_9
B_0 B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8 B_9
Embed Position 2:
Embed Position 3:
Embed Position 1:
A_0 A_1 A_2 A_3 A_4 A_5 A_6 A_7 A_8 A_9 A_{10} A_{11}
B_0 B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8 B_9 B_{10} B_{11}
Embed Position 2:
Embed Position 3:
Embed Position 4:

(a) (b)

Figure 5.5. Embedding location groups proposed for puncture code rates: (a) R_{equiv} = 10/14 embedding positions and (b) R_{equiv} = 12/15 embedding positions.



(a) (b)

Figure 5.6. Evaluation of embedding location for punctured convolutional codes; BPSK modulation, 100 trials over AWGN: (a) R_{equiv} = 10/14 embedding on base $R = 2/3$ code and (b) R_{equiv} = 12/15 embedding on base $R = 3/4$ code.

returned a minor improvement in BER, when this embedding location was implemented in simulations for 802.11ac, it was found that the second embedding location returned improved PER performance. This change in performance was found to be associated with

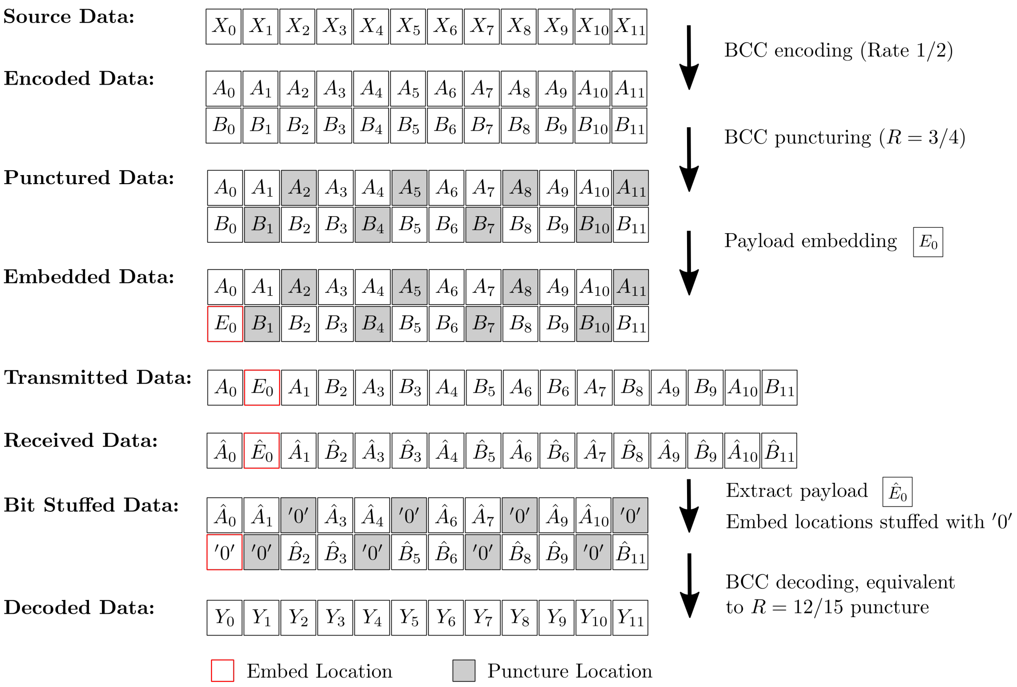


Figure 5.7. Proposed embedding scheme for punctured convolutional codes conducted on $R = 3/4$ code resulting in $R_{equiv} = 12/15$ code

the characteristic of the Viterbi decoder utilized by each implementation. For our initial simulations in the MCS model testbed, a continuous mode Viterbi-decoder was specified where the encoder state is retained between each frame; conversely, the 802.11ac simulation utilized a truncated decoder where the encoder resets after each frame. A depiction of the proposed embedding technique with the experimentally derived embedding location is shown in Figure 5.7. From this structure, we developed the concept of an embedding interval, ν, which describes the length of the embedding period. For the case developed for $R_{equiv} = 10/14$, with $m = 5$ appended puncture periods and $n = 3$ as the number of output bits contained in each puncture period, $\nu = m \times n = 5 \times 3 = 15$. For the $R_{equiv} = 12/15$ embedding with $m = 4$ and $n = 4$, $\nu = 16$.

5.3 Forward Error Correction of Embedded Message

The selection of a FEC code for the embedded payload will not only impact the overall embedding capacity but impact the error rate of the received payload. One area briefly mentioned in Chapter 3 was the concept of unequal error protection between the embedded data and the underlying communications channel. Under normal circumstances, the FEC selected to protect the embedded data should provide at least as much redundancy as the underlying communications channel. Due to the unique nature of these embedded channels, and the fact that some implementations are likely simplex in nature and lack the ability to utilize ARQ-like mechanisms, it may be advantageous to select a more robust FEC code than the underlying channel.

An additional case exists within adaptive rate communication systems if the channel state is known to exceed the requirements for the current MCS; this situation would generally exist if a decremented MCS was intentionally selected in an attempt to increase the available embedding capacity. In this case, a higher-rate FEC code can be selected with the knowledge that the channel conditions are more favorable than indicated by the MCS index.

To get a sense of the impact of FEC selection on the error performance of the embedded payload we took advantage of the analytical expressions that exist for our BPSK and BCC-based MCS model. An upper bound on the resulting probability of packet error, P_p can then be determined by [79]

$$P_p < 1 - (1 - P_e)^{8L} , \tag{5.7}$$

where P_e is the union bound on the error event probability and L is the length of the packet in octets. As opposed to the P_b bound calculated in (5.4), which accounts for the fact that each error on the decoding path will result in multiple information bit errors, P_e simply returns the probability of any error event occurring. Similar to our earlier calculation of P_b, the distance spectra for our codes can be utilized [63]

$$P_e < \sum_{d=d_{free}}^{\infty} a_d P_d , \tag{5.8}$$

where a_d is the event weight and P_d remains the pairwise error probability for the selected modulation and channel. Combining (5.3), (5.7), and (5.8) yields the following expression for the P_p for BPSK in AWGN

$$P_p < 1 - \left[1 - \left(\sum_{d=d_{free}}^{\infty} a_d Q \left(\sqrt{2d \frac{E_b R}{N_0}} \right) \right) \right]^{8L_p}, \tag{5.9}$$

with L_p representing the length of the PSDU in octets.

To get a sense of the estimated error performance based on the selection of varying FEC codes we utilized the expressions for the upper bounds of P_p from (5.9) to determine E_b/N_0 at a designated threshold; in this case we utilized a value of $P_p = 0.1$. The values listed in Table 5.4 represent the minimum expected performance for the communication system operating at the designated MCS.

Table 5.4. Required E_b/N_0 for MCS indices to maintain $P_p = 0.1$; BPSK with 4096-octet PSDU in AWGN.

MCS Index	**Coding Rate**	$\boldsymbol{E_b/N_0}$
A	$R = 1/2$	4.084 dB
B	$R = 2/3$	4.681 dB
C	$R = 3/4$	5.247 dB
D	$R = 5/6$	5.889 dB

We used the E_b/N_0 for each MCS index to estimate the probability of bit error, $\hat{P}_b$, for the embedded payload using (5.5) with the results being recorded in Table 5.5.

Table 5.5. Estimated upper bound for P_b of embedded data when $P_p = 0.1$; BPSK MCS model with 4096-octet PSDU in AWGN.

Selected FEC for	$\boldsymbol{\hat{P}_b}$ **of Embedded Bits at** $\boldsymbol{P_p = 0.1}$			
Embedded Data	MCS A	MCS B	MCS C	MCS D
$R = 1/2$	1.35×10^{-5}	1.53×10^{-6}	1.62×10^{-7}	9.50×10^{-9}
$R = 2/3$	7.01×10^{-5}	7.64×10^{-6}	8.74×10^{-7}	6.26×10^{-8}
$R = 3/4$	4.11×10^{-4}	4.77×10^{-5}	6.67×10^{-6}	6.60×10^{-7}
$R = 5/6$	3.9×10^{-3}	3.19×10^{-4}	4.16×10^{-5}	4.66×10^{-6}

The lower-left corner of the table, highlighted in grey, represents FEC selections that will deliver a higher embedding capacity but provide less error protection than the underlying communication system.

5.4 Embedding Capacity Estimation

To estimate the embedding capacity of our proposed techniques for convolutional codes, it is first necessary to extend the concept of the embedding interval, ν, to our implementation for unpunctured codes. For the unpunctured case, the embedding is conducted in a manner consistent with an equivalent puncture code, R_{equiv}; using our standard definition of a punctured code

$$R_{equiv} = \frac{k_{equiv}}{n_{equiv}} \tag{5.10}$$

where n_{equiv} represents the number of output bits and k_{equiv} is the number of input bits (as well as the puncture period of the equivalent code rate). Using these values, and the rate of the parent convolutional code, R_P, it is possible to derive an expression for the embedding interval of the unpunctured implementation, ν_U, where

$$\nu_U = \frac{k_{equiv}}{R_P} \ . \tag{5.11}$$

Next, we must define the number of embedded bits in each embedding interval, b_ν. For our proposed implementation on punctured code rates, $b_\nu = 1$. For the unpunctured case, $b_{\nu,U}$ is related to k_{equiv} of the equivalent puncture rate and the parent code rate

$$b_{\nu,U} = \frac{k_{equiv}}{R_P} - n_{equiv} = \nu_U - n_{equiv} \ . \tag{5.12}$$

Now that we can express the performance of both embedding methods in terms of an embedding interval, we can develop a common equation to determine the raw embedding capacity of each approach. For the following equations, we will use the notation ν and b_ν for the embedding interval and capacity per embedding interval; while we have defined ν and b_ν for embedding in MCS that employ punctured codes, the equations will be equally valid for ν_U and $b_{\nu,U}$ from the unpunctured techniques. Given the length of the underlying data block in octets, L_P, and the rate of the convolutional encoder used to support the MCS of the underlying communication channel, R_C, it is possible to determine the number of coded bits per block, N_{CBPB}, where

$$N_{CBPB} = \left\lceil \frac{8L_P}{R_C} \right\rceil , \tag{5.13}$$

which can then be combined with ν and b_ν to determine an expression for the raw embedding capacity of our implementations where $C_{E,blk}$ is the raw embedding capacity measured in bits-per-block

$$C_{E,blk} = b_\nu \left\lfloor \frac{N_{CBPB}}{\nu} \right\rfloor \ . \tag{5.14}$$

We used this raw capacity to determine the size of the embedded payload, L_E, when R_{EC} is the FEC code rate selected to protect the embedded bits

$$L_E = \lfloor C_{E,blk} R_{EC} \rfloor \ . \tag{5.15}$$

Combining (5.13), (5.14), and (5.15) provides an expression for the capacity of our convolutional embedding methods, expressed as the size of the FEC-protected embedded payload and as a function of the underlying data block and the convolutional code used by the current MCS

$$L_E = \left\lfloor b_\nu R_{EC} \left\lfloor \frac{1}{\nu} \left\lceil \frac{8L_P}{R_C} \right\rceil \right\rfloor \right\rfloor . \tag{5.16}$$

5.5 Embedding within IEEE 802.11ac VHT

Once our proposed block embedding technique had been validated via simulation in our BPSK testbed, we worked to adapt our embedding technique within a MATLAB implementation of 802.11ac VHT. Our embedding testbed was once again adapted from a MATLAB-developed script to measure PER; embedding and extraction of the hidden message required modifications to existing encode and decode functions used within the MATLAB WLAN Toolbox. While we did embed randomly generated payload bits at the transmitter, and subsequently extract them at the receiver, the only metric actively collected during our VHT embedding trials was the PER of the underlying communication channel. Based on earlier observations, our primary concern in these trials was the performance impact on the carrier at each embedding interval.

Our simulation implemented the transmission of a single VHT PPDU and we evaluated the performance of our system based on a 10% PER threshold established by the receiver validation criteria outlined in [44]. While this performance threshold was specified for a PSDU of 4096 octets, we discovered that in the MATLAB implementation of 802.11ac, there is no method to directly set the length of the PSDU. This change in implementation is due to the fact that in VHT all frames are considered to be aggregate frames, even if they only contain a single MAC Protocol Data Unit (MPDU) [81]. Instead, we must designate the length of the aggregate MPDU (A-MPDU) which is then encapsulated by the simulated MAC layer to become the PSDU.

MATLAB faithfully implements this portion of the protocol, but as a result, the length of the PSDU will vary between MCS rates even if the length of the A-MPDU remains unchanged. The amount of overhead in this encapsulation is inconsistent between MCS rates as it is related to the number of data bits per symbol, N_{DBPS}, which is determined by the selected modulation type and coding rate as well as other characteristics including the bandwidth (BW), number of spatial streams, N_{SS}, and order of the MIMO implementation. As an example, we opted to conduct all of our VHT trials using 8 × 8 MIMO, 8 spatial streams, and a BW of 80 MHz. To determine the number of data bits per symbol, it was first necessary to calculate N_{CBPS}, or the number of coded data bits per symbol [36]

$$N_{CBPS} = N_{SD} \times N_{FS} \times N_{SS} \times N_{BPSCS} \ , \tag{5.17}$$

where N_{FS} is the number of frequency segments, N_{SD} is the number of complex data numbers for each segment, N_{SS} is the number of spatial streams, and N_{BPSCS} is the number of coded data bits per subcarrier per spatial stream [36]. The values of N_{BPSCS}, N_{FS}, N_{SS} and N_{SD} are all functions of the selected MCS and can be found in [44].

The results from (5.17) can then be used to determine the number of data bits per symbol [36],

$$N_{DBPS} = N_{CBPS} \times R \tag{5.18}$$

where R is the code rate of the selected MCS. For our selected parameters, the values of N_{DBPS} for each MCS index are displayed below in Table 5.6, along with the number of symbols, N_S, required to transport an A-MPDU length of 4096-octets, and the resulting size of PSDU, L_{PSDU}, after accounting for any required overhead. Despite variations in PSDU length, we elected to conduct all of our simulations with a consistent A-MPDU as it was an accurate reflection of the variations that would occur between adjacent MCS in real-world VHT implementations.

Table 5.6. Number of databits per symbol, number of required symbols, and associated PSDU length for MCS indices in 802.11ac; 8×8 MIMO, 8 spatial streams, BW of 80 MHz and A-MPDU of 4096 octets.

MCS	N_{DBPS}	N_S	L_{PSDU}
1	1872 bits	18	4207 octets
2	2808 bits	12	4208 octets
3	3744 bits	9	4208 octets
4	5616 bits	6	4207 octets
5	7488 bits	5	4675 octets
6	8424 bits	4	4207 octets
7	9360 bits	4	4673 octets
8	11232 bits	3	4205 octets
9	12480 bits	3	4673 octets

Before conducting any embedding trials, it was first necessary to conduct baseline performance testing of the unembedded VHT MCS indices; as before, the performance was evaluated as PER across a range of SNR points. The results of our initial validation of MCS indices 1 through 9, conducted over an AWGN channel with an A-MPDU length of 4096 octets, are shown in Figure 5.8.

Similar to the technique utilized in 802.11ad, embedding of the desired payload would take place following the encoder and prior to the modulator; extraction would take place following the demodulator and prior to the decoder. Conducting this process within VHT adds an extra layer of complexity due to the presence of multiple data streams in the encoder.

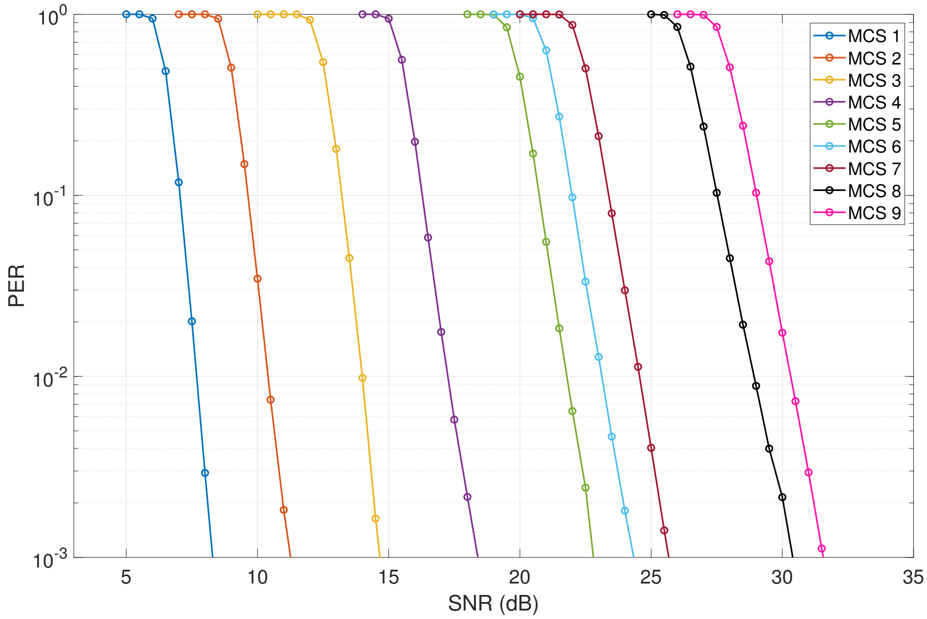


Figure 5.8. PER versus SNR performance curves for 8×8 MIMO IEEE 802.11ac VHT modulation and coding schemes with BW of 80 MHz; performance average of 10000 trials over AWGN channel with single 4096-octet A-MPDU.

After the padded data field of the PPDU is passed through the scrambler, it is parsed into multiple data streams before being passed to the BCC encoders. The multiple encoders are necessary due to encoder data rate limitations of 600 Mbps [36]. The number of data streams, N_{ES}, and therefore the number of parallel BCC encoders and decoders for a specified MCS can be determined based on the value of N_{DBPS} previously calculated in (5.18) [36]

$$N_{ES} = \left\lceil \frac{N_{DBPS}}{600 \times 3.6} \right\rceil \tag{5.19}$$

where 3.6 is a constant that represents the short GI duration, and 600 is the data rate limitation for the BCC encoder measured in Mbps.

For our simulated VHT embedding trials, we opted to utilize a BW of 80 MHz and $N_{SS} = 8$; these characteristics established the values of $N_{SD} = 234$ and $N_{FS} = 1$. For an MCS-index of 5, which has $R = 2/3$ code, and 64-QAM modulation with $N_{BPSCS} = 6$, we were able to determine from (5.17) and (5.18) that $N_{DBPS} = 7488$. Passing this value through (5.19) results in a requirement for four BCC encoders.

To implement our proposed embedding scheme, payload bits were embedded into each of the N_{ES} data streams at the specified embedding interval, ν, after the underlying data stream was encoded and punctured, but prior to additional stream parsing and interleaving. At the receiver, the embedded payload must be recovered after de-interleaving and prior to the N_{ES} data streams entering the BCC decoders. The method used to prevent the bit locations that carry the embedded data from being considered by the BCC decoders are dependent on implementation and would have to be evaluated on a case-by-case basis. In our simulated VHT trials, we were to able obtain consistent results by either replacing the bit locations with LLR = 0, or modifying the logic within the decoder to identify the embedded bit locations as containing invalid information.

Given information about the size of the parsed PPDU data field in bits, L_{DS}, which represents the length of the uncoded data that passed through each of the N_{ES} BCC encoders, we can modify (5.14) and (5.15) to determine the length of the FEC-protected embedded payload, $L_{E,VHT}$, carried in each PPDU

$$L_{E,VHT} = N_{ES} \left\lfloor b_\nu R_{EC} \left\lfloor \frac{1}{\nu} \left\lceil \frac{L_{DS}}{R_C} \right\rceil \right\rfloor \right\rfloor, \tag{5.20}$$

where R_C is the MCS code rate, R_{EC} is the rate of the FEC code used on the embedded payload, and ν is the designated embedding interval.

5.6 Implementation of TGac Fading Channel

To investigate the performance of these embedding techniques under more realistic channel conditions required the use of an appropriate fading model. The MATLAB WLAN Toolbox contains a system object that implements multipath fading based on channel model characteristics accepted by IEEE P802.11 Task Group ac (TGac) as outlined in [82]. This model is itself an extension on the earlier Task Group n (TGn) channel model with modifications to account for the improvements implemented in 802.11ac to enable higher throughput including the use of higher order MIMO, increased BW, and the implementation of multi-user MIMO (MU-MIMO).

The MATLAB implementation of the TGac fading model allows the selection of one of six delay profiles described in [83] that were selected to represent different WLAN channel environments; we selected *Model-D* which was developed to represent a typical office environment [84]. Key attributes associated with this delay profile include the root mean squared (rms) delay spread of 50 nanoseconds (ns), maximum delay of 390 ns, Rician K-factor of 3 dB, and a break-point distance of 10 meters. [84]. The break-point distance of the model specifies the distance between stations at which the channel would be considered NLOS. As a result, the distance between the transmitter and receiver impacts both the computation of path loss as well as the transition between LOS and NLOS conditions.

5.7 Summary

In this chapter, we developed the concept for embedding within convolutional codes and presented techniques that could be implemented depending on whether a convolutional code was the parent code, or an already punctured version. These techniques were initially validated within a theoretical MCS model that was specifically developed to allow the simulated results to be validated by performance bounds derived from analytical expressions. We explored the trade-offs involved with the selection of FEC techniques to protect our embedded payload and derived capacity equations to determine the amount of data that could be carried as an embedded payload based on the parameters of the underlying communication system and the embedding implementation. Finally, we discussed the multipath fading model that would be utilized to explore the performance of our embedding techniques under more realistic channel conditions. Relevant results obtained from the simulations developed for these convolutional embedding techniques will be presented in Chapter 6.

CHAPTER 6: Embedding Simulation Results

Following the development of error correction code-based implementations for both LDPC and convolutional codes, we conducted extensive MATLAB simulations to evaluate their performance in terms of both embedding capacity as well as impact to the underlying communication system. MATLAB versions 2018b and 2019a were utilized to generate the results contained in this chapter along with functions contained within the MATLAB WLAN Toolbox. Existing end-to-end simulations, developed and published by MATLAB as part of WLAN Toolbox documentation, were used to develop our 802.11ac and 802.11ad testbeds. We compared these existing scripts against [44] as well as other relevant references [36], [51], [53], [59], [63] to validate the critical functional components of the encoding, modulation, demodulation, and decoding process.

We present the results of our simulations in three distinct sections. The first section characterizes the performance of the LDPC-based embedding within the IEEE 802.11ad DMG standard. The second section presents the results of our convolutional code embedding in both the original BPSK model as well simulated trials conducted for IEEE 802.11ac VHT. The final section addresses the performance of both DMG and VHT embedding under simulated multipath fading environments in an attempt to validate our embedding techniques under more realistic channel models.[6, 7]

6.1 Embedding in LDPC Codes

Initial simulations to assess the performance of the proposed embedding scheme within the LDPC codes of the 802.11ad DMG standard were conducted against an AWGN environment. The first series of trials presented in this section represents the embedding case where the MCS of the underlying channel is intentionally decremented; in this case, we can assume

[6] Portions of this chapter were previously published by IEEE [5]. Reprinted, with permission, from P. M. B. Harley, M. Tummala and J. C. McEachen, “High-Throughput Covert Channels in Adaptive Rate Wireless Communication Systems,” *2019 International Conference on Electronics, Information, and Communication (ICEIC)*, Auckland, New Zealand, 2019, pp. 1-7.

[7] Portions of this chapter were used in an upcoming paper submission for the 53rd Hawaii International Conference on System Sciences (HICSS), slated for January, 2020.

that the channel conditions are sufficient to support the next higher MCS index, I_{C+1}. As a result, each LDPC codeword can be embedded to the maximum extent so long as the performance of the embedded channel operating at MCS index I_C is able to maintain the designated PER threshold at a SNR equivalent or lower than that required by I_{C+1}. In the second series, we examined the case where we do not decrement the MCS prior to embedding. Instead, we sought to evaluate the relationship between the available margin of embedding, M_E, and the embedded channel capacity, C_{CW}, measured in bits-per-codeword.

6.1.1 Decremented Modulation and Coding Scheme Embedding

Our first MATLAB simulations served as a proof of concept of the embedding technique and were designed to address three main objectives [5]. First, determine if a specified payload could be successfully embedded and subsequently extracted from an LDPC codeword. Second, identify the upper limit of embedding, measured in bits-per-codeword, which would result in a noticeable performance degradation to the underlying communication system [5]. Finally, we sought to assess whether the embedded payload could be reliably estimated given the presence of channel noise [5].

Based on our original examination of the implementation of the IEEE 802.11ad DMG protocol, initial embedding simulations were conducted at MCS 9 and evaluated under channel-conditions that would support MCS 9.1. The simulated transmitter replaced the first 48 parity bits of each $R = 13/16$ LDPC codeword with a unique payload; the embedded data for these initial trials was not protected by any error correction mechanisms. After determining that the embedding technique was feasible, additional trials were then conducted to determine the behavior of the embedded channel, and the underlying communication system, at different levels of embedding [5]. Decrementing the selected MCS index resulted in a reduction in the maximum data rate of the underlying communications system; for example, when MCS 9 is selected in lieu of MCS 9.1, the maximum data rate is reduced from 2695 Mbps to 2502.5 Mbps [44] [5].

Results of simulated trials where the PER for varying embedding rates are compared to that of the baseline MCS index are displayed in Figure 6.1; as expected, the PER of MCS 9, at an embedding rate of 48 bits-per-codeword, is equivalent to that of baseline MCS 9.1 [5]. The embedded data during these initial trials was uncoded and therefore did not benefit

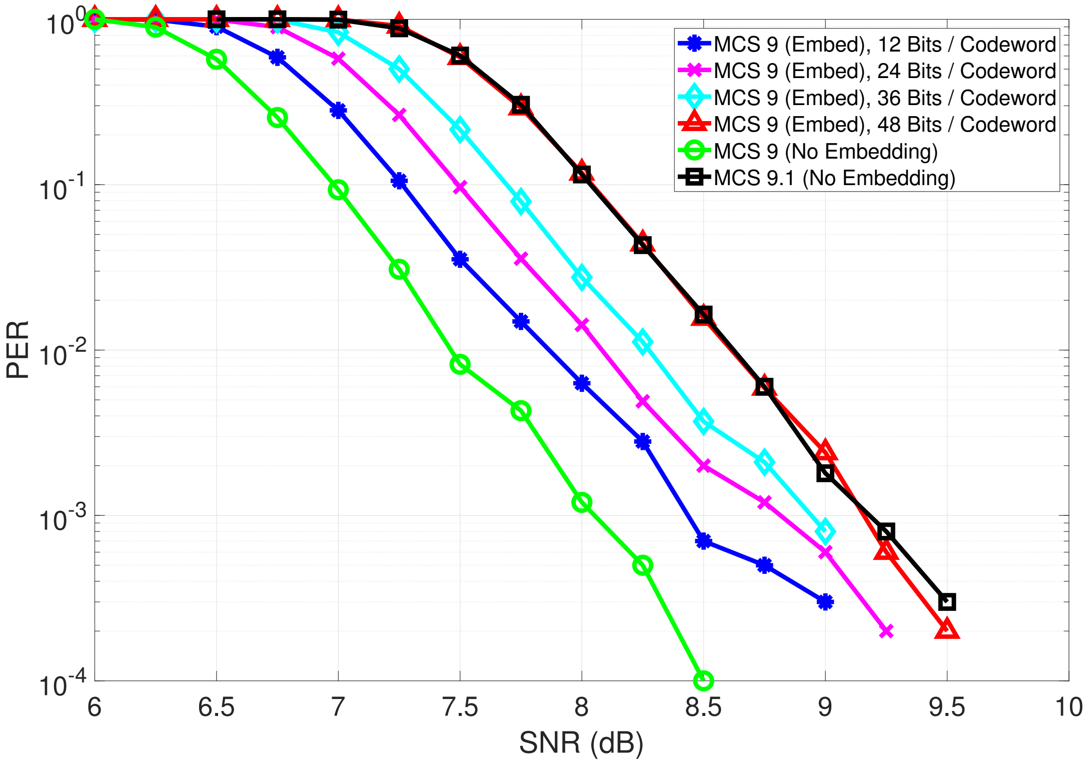


Figure 6.1. Packet error ratio of underlying communication channel; MCS 9, no FEC, 10000 PSDU per SNR. DMG PHY simulation under AWGN channel. Source: [5], © 2019 IEEE.

from any error correction [5]. As a result, the BER of the received embedded payload was consistent with the probability of bit error, P_b, of uncoded QPSK [5]. The comparison of the embedded BER to the P_b for QPSK is shown in Figure 6.2.

A surprising observation from this original set of trials related to the fact that the performance of the underlying communications channel, as characterized by PER, appeared to be relatively predictable when subjected to varying levels of embedding activity [5]. As illustrated in Figure 6.1, the PER of the underlying communication system increased up to the limit of 48 bits-per-codeword, with embedding rates of 24 bits-per-codeword requiring a lower SNR to achieve a given PER than the 36 bits-per-codeword embedding rate, but a higher SNR than the 12 bits-per-codeword rate [5]. The monotonic relationship between increased SNR and increased embedding capacity formed the foundation for our proposed extension to variable rate embedding without the need to decrement the MCS.

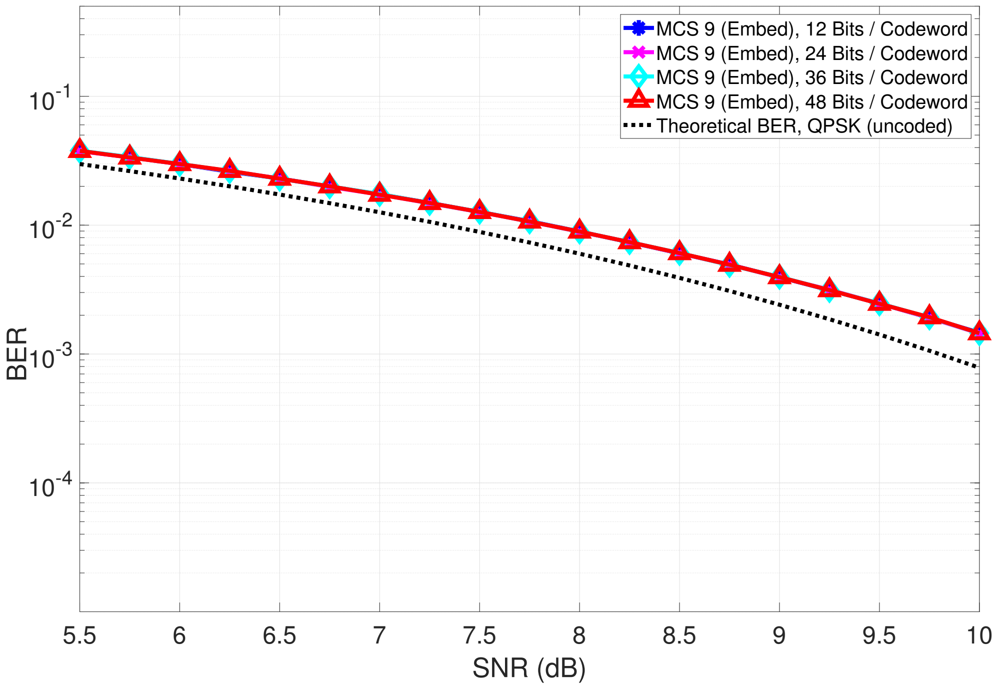


Figure 6.2. Bit error ratio for received hidden message; MCS 9, no FEC, 10000 PSDU per SNR. DMG PHY simulation under AWGN channel. Source: [5], © 2019 IEEE.

After we completed our initial trials at MCS 9, we repeated similar trials on the 802.11ad MCS indices that utilized QPSK modulation. The maximum number of bits-per-codeword for each 4096-octet PSDU was determined experimentally for each MCS [5]. We utilized these results to calculate the ratio of embedded data to the amount of data bits contained in each PPDU (including padding); we then multiplied this ratio against the data rates published in [44] to estimate the embedded data rate at each MCS. Embedding capacity estimates are displayed in Table 6.1.

Of note, the embedding rates per codeword from these trials align well with the theoretical capacity based on the difference in redundancy discussed in Section 3.4. A comparison between the predicted bounds and the results of the simulated trials are shown in Table 6.2; in all cases, our simulated results met or exceeded the predicted embedding rates. For the cases where the embedding capacity exceeded the difference in redundancy between adjacent MCS, we believe that the iterative decoding nature utilized for LDPC codes provided additional coding gain that supported a higher than expected embedding rate. Although the

Table 6.1. Summary of results: DMG, single carrier, QPSK modulation, 4096-octet PSDU, no FEC

MCS Index		**LDPC Code Rate**	**Max Embed per CW**	**Embedded Bits**		
Dictated by Channel Conditions	Used for Embedding			Per 4096-octet PSDU	As % of Overt Data	Estimated Data Rate (Mbps)
7	6	1/2	95 bits	9310 bits	28.27%	435.36
8	7	5/8	85 bits	6715 bits	20.24%	389.62
9	8	3/4	48 bits	3168 bits	9.52%	219.91
9.1	9	13/16	48 bits	2928 bits	8.79%	220.00

lack of FEC in these trials maximized the embedding capacity of the information-hiding technique, the high BER of the embedded payload would not have facilitated a reliable embedded communications channel [5].

Table 6.2. Predicted embedding rates, measured in bits per LDPC codeword, compared to simulated results, 802.11ad DMG SC QPSK MCS

MCS Index	**Predicted Embedding Rate**	**Simulated Results**
6	84	95
7	84	85
8	42	48
9	48	48

6.1.2 Forward Error Correction of Embedded Message

For the initial trials conducted at MCS 9, the maximum embedding capacity under channel conditions that could support MCS 9.1 was 48 bits-per-codeword; using (4.3), the raw embedding capacity was determined to be 2928 bits for a 4096-octet PSDU. Selecting the least robust FEC code from the 802.11ad standard, the punctured $R = 7/8$ code, it was only possible to embed four 624-bit codewords in each PSDU, leaving 240 bits unaltered. From (4.7), it was possible to calculate that the FEC-protected payload in each 4096-octet PSDU was therefore reduced to just 2184 bits.

Although capacity was reduced, FEC significantly improved the embedded payload BER. The SC DMG specification calls for a minimum PER of 10^{-2}, or 1% [44]; as shown in Figure 6.1, a 1% PER is achieved at an SNR of approximately 8.6 dB [5]. At 8.6 dB, the BER for the FEC-protected embedded payload, illustrated in Figure 6.3, is approximately 7.8×10^{-7} [5]. As before, trials were then conducted for all QPSK-based MCS with embedding capacity estimates provided in Table 6.3 [5].

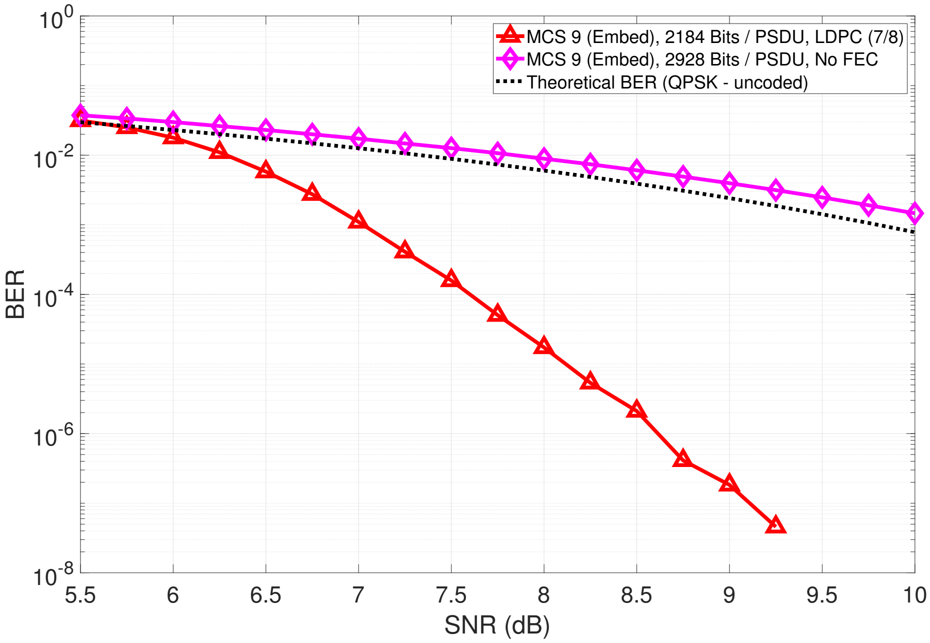


Figure 6.3. Bit error ratio for received hidden message; MCS 9, LDPC(7/8) FEC, 10000 PSDU per SNR. DMG PHY simulation under AWGN channel. Source: [5], © 2019 IEEE.

Table 6.3. Summary of results: DMG, single carrier, QPSK modulation, FEC applied to embedded hidden data.

MCS Index		**LDPC Code Rate**		**Embedded Bits**		
Dictated by Channel Conditions	Used for Embedding	Overt Data	Embedded Data	Per 4096-octet PSDU	As % of Overt Data	Estimated Data Rate (Mbps)
7	6	1/2	5/8	5460 bits	16.58%	255.36
8	7	5/8	3/4	4536 bits	13.67%	263.16
9	8	3/4	13/16	2184 bits	6.57%	151.67
9.1	9	13/16	7/8	2184 bits	6.56%	164.10

6.1.3 Interleaving of Embedded Data

An additional series of trials was then conducted utilizing the interleaved embedding technique with the aim of reducing the distortion on the underlying communications channel without sacrificing the FEC-protected payload capacity of the embedded channel.

While the modification did not result in an increase in embedding capacity, it did consistently reduce the impact of the embedding on the underlying communication system [5]. A representation of this improvement can be seen in Figure 6.4 where the PER for interleaved embedding is shown alongside the standard FEC implementation; in both cases, 2184 message bits were embedded in each 4096-octet PSDU but the interleaved method achieved this performance with a 0.25 dB reduction in the required SNR [5]. As shown in Figure 6.5, the embedded data BER for the interleaved case remains unchanged from the standard FEC implementation [5].

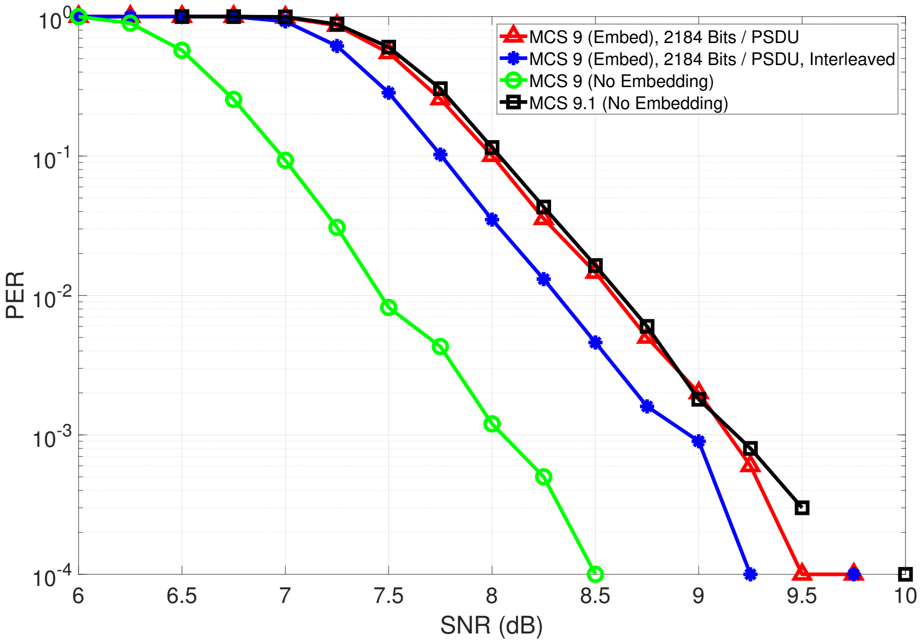


Figure 6.4. Packet error ratio of underlying communication channel; MCS 9 with interleaved embedding, LDPC(7/8) FEC, 10000 PSDU per SNR. DMG PHY simulation under AWGN channel. Source: [5], © 2019 IEEE.

This improved performance remained consistent across the previously considered MCS indices that utilize QPSK modulation [5]. A summary of the SNR performance of this final iteration, along with the associated BER for the embedded data, is shown in Table 6.4 [5].

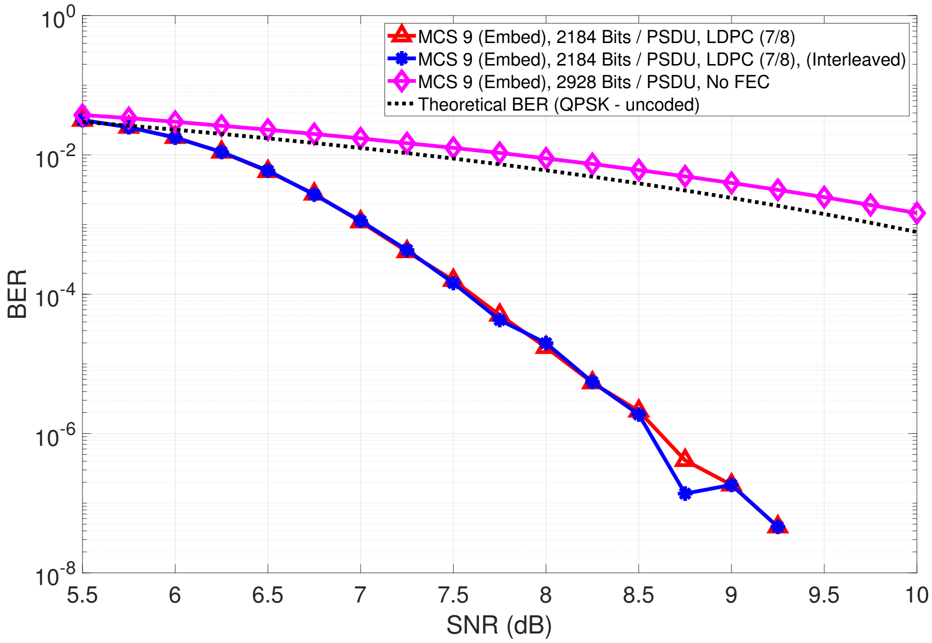


Figure 6.5. Bit error ratio for received hidden message; MCS 9 with interleaved embedding, LDPC(7/8) FEC, 10000 PSDU per SNR. DMG PHY simulation under AWGN channel. Source: [5], © 2019 IEEE.

Table 6.4. Summary of results: DMG, single carrier, QPSK modulation, FEC and interleaving applied to embedded hidden data

MCS Index		**SNR Required at 1% PER**		**Embedded Bits per 4096-octet PSDU**	**Embedded Data BER (at 1% PER)**
Dictated by Channel Conditions	Used for Embedding	Dictated by Channel Conditions	Used for Embedding		
MCS 7	MCS 6	5.07 dB	4.93 dB	5460 bits	5.3×10^{-6}
MCS 8	MCS 7	6.41 dB	6.26 dB	4536 bits	5.4×10^{-6}
MCS 9	MCS 8	7.47 dB	7.21 dB	2184 bits	4.1×10^{-6}
MCS 9.1	MCS 9	8.61 dB	8.31 dB	2184 bits	7.8×10^{-7}

6.1.4 Embedding within Existing MCS

Using the analytical tools and techniques outlined in Section 4.4, we sought to fully characterize the performance of our LDPC embedding scheme for IEEE 802.11ad in the AWGN environment. The goal of these simulations was to gather sufficient data to summarize the error performance of the underlying communication system over the full range of embedding rates; this data would then be used to develop a function to describe the maximum level of embedding that could be conducted at a given channel state.

As the focus of these trials was to determine the performance of the underlying system, we did not evaluate the BER performance of the embedded data. The embedding trials in Section 6.1.2 clearly established that the application of FEC techniques to the embedded payload were able to deliver acceptable error performance at even the highest embedding rates. While the use of FEC, and the requirement to transmit parity bits, will reduce the size of the embedded payload that can be carried in each codeword, this reduction can be easily calculated based on the selected FEC code rate and equations presented in Section 4.3.1.

Expanded embedding trials were conducted for all SC MCS with the exception of MCS 1 and MCS 5. MCS 1 was excluded as it is the only SC MCS that utilizes a repetition factor ($\rho = 2$); MCS 5 was excluded because it is outperformed by MCS 6 under the same channel conditions. Due to the fact that MCS 6 offers a higher data rate and superior error performance, not only is MCS 5 unlikely to be selected by a link adaptation algorithm, but the MCS would also be a poor candidate for FEC-based embedding. A plot of the results from the embedding trials conducted for the SC QPSK MCS is shown in Figure 6.6.

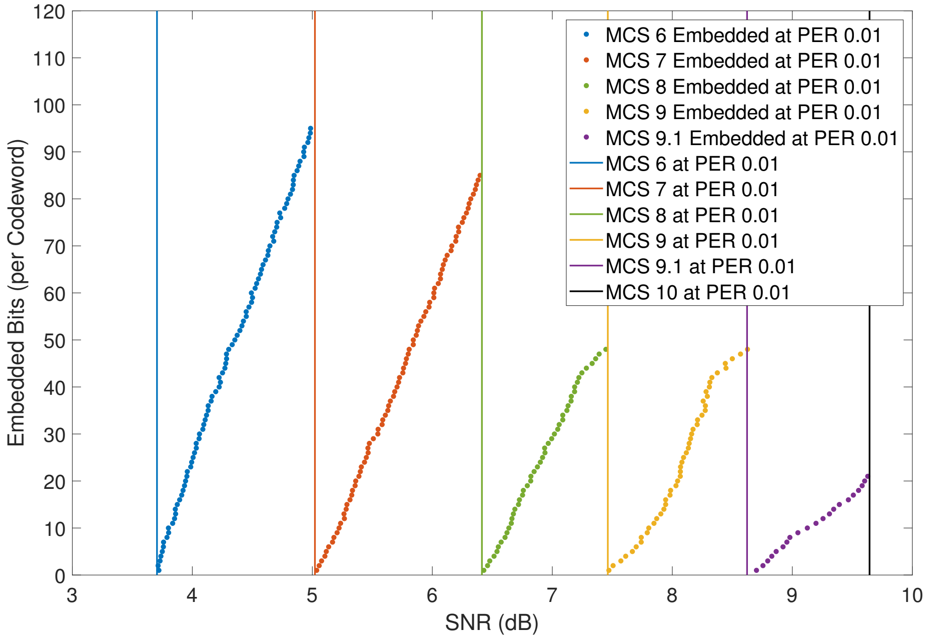


Figure 6.6. Embedded bits-per-codeword versus SNR for all 802.11ad $\pi/2$-QPSK modulated MCS indices; embedding conducted in first n parity bits of each LDPC codeword

When the regression lines obtained for each MCS were plotted against the simulated results, the fit was generally very good. That said, as shown in Figure 6.7, there were a number of points, highlighted by red circles, where these regression lines deviate from the simulated results. For the SC QPSK MCS, the differences were particularly noticeable as the regression line approached the maximum embedding rate for MCS 8 and at both the maximum and minimum embedding rates for MCS 9. In these cases, alternative bounds were identified to adjust the original regression lines to ensure the estimated capacity was achievable at the given SNR and within the required PER. The modified slope was then used to determine the associated estimated embedding coefficient, $\hat{r}_E$; similar discrepancies in the remaining SC MCS were similarly identified and corrected. The estimated embedding coefficient for all SC MCS are recorded in Table 6.5.

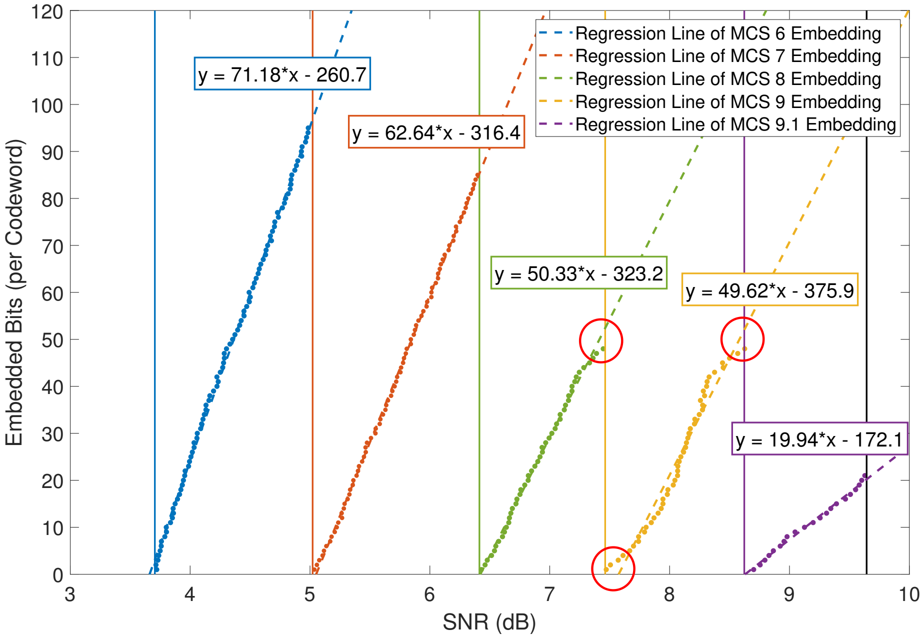


Figure 6.7. Regression lines of embedded bits per codeword versus SNR for all 802.11ad QPSK modulated MCS

For MCS 8 and 9, we noted that the deviation from the original regression line was particularly pronounced as the result of a rate change that took place after the 42-bit-per-codeword embedding rate. Since the cycle permutation matrix used to construct the $\boldsymbol{H}$ for 802.11ad has a dimension of 42×42, we investigated the structure of $\boldsymbol{H}$ in this location. We determined that for both LDPC matrices, the transition into an adjacent submatrix resulted in embedding locations 43 to 48 contributing to check nodes that were already associated with embedded positions. Having multiple locations contributing an LLR = 0 to the first stage of the decode process ($\eta^{[1]}$), would result in all locations in the resulting matrix, including the embedded bit positions, being set to zero. As a result, during the first bit node update, the impacted check nodes would not contribute any information toward recovering the original values of the embedded locations. This had the impact of effectively reducing the equivalent column weight, w_c, in these bit positions during the first cycle. For the $R = 13/16$ code in MCS 9, this issue was compounded by the fact that $w_c = 2$ at the embedding locations 43 through 48 as opposed to $w_c = 3$ for each of the first 42 embedding locations.

Table 6.5. Estimated embedding coefficient, $\hat{r}_E$, for 802.11ad (first n Parity Bits)

Modulation		**Rate**		**Rate**		**Rate**		**Rate**		**Rate**
	MCS	1/2	**MCS**	5/8	**MCS**	3/4	**MCS**	13/16	**MCS**	7/8
$\pi/2$-BPSK	2	78.62	3	66.14	4	46.86	5	N/A	–	–
$\pi/2$-QPSK	6	71.18	7	62.64	8	46.47	9	41.27	9.1	19.94
$\pi/2$-16QAM	10	57.40	11	54.88	12	43.52	12.1	39.02	12.3	19.36
$\pi/2$-64QAM	–	–	12.3	44.02	12.4	37.08	12.5	36.72	12.6	20.52

As shown in Table 6.5, the value of $\hat{r}_E$ generally decreased with increased code rate or order of modulation. This result aligns with our intuition related to FEC-based embedding; $\hat{r}_E$ is a measure of the estimated embedding capacity per codeword for every additional dB of SNR. Therefore, as the modulation becomes more complex or the redundancy of the FEC is reduced, we would expect to see a decrease in the ability to embed data without increasing the PER. The most significant reduction in embedding performance is observed for the MCS that utilize the $R = 7/8$ codes; this extremely poor performance is due to the fact that the $R = 7/8$ code is itself a punctured version of the $R = 13/16$ LDPC code.

Once we had completed the trials for all SC MCS, we repeated all trials for an alternate embedding position within the LDPC codewords. Initial trials identified strong performance when embedding was conducted at the end of the data section of the systematic LDPC codeword. The results of embedding in the last n data bits for the QPSK MCS indices is shown in Figure 6.8; it is clear that this embedding location resulted in a steeper slope for all MCS with the exception of MCS 9.1. This steeper slope corresponds to a larger $\hat{r}_E$, and a higher estimated embedding capacity.

Embedding trials were repeated for all SC MCS with the exception of MCS 1 and

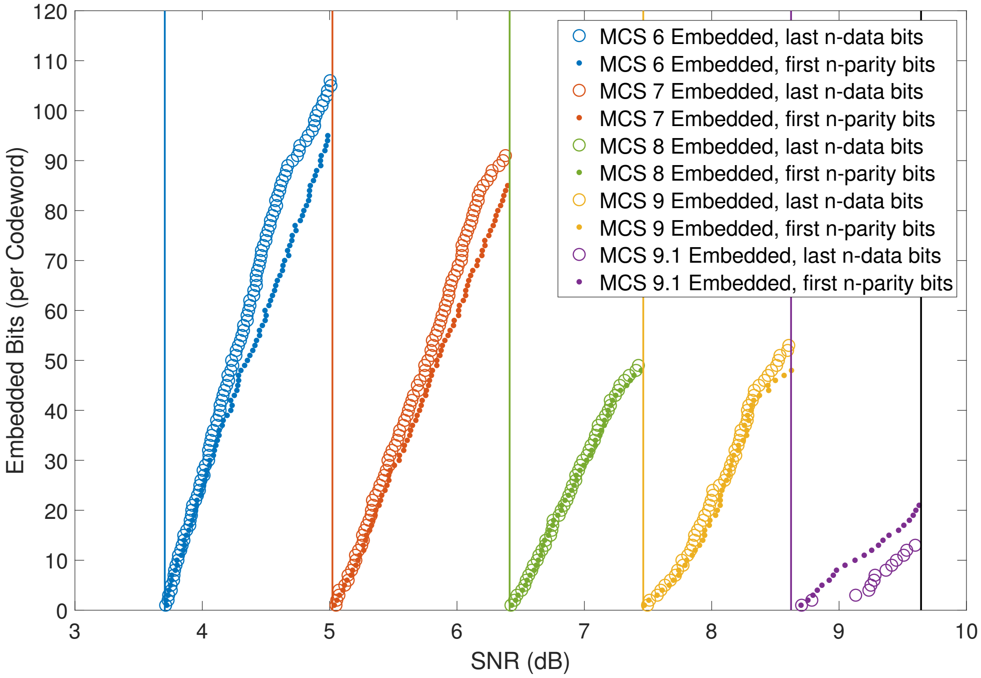


Figure 6.8. Comparison of embedding capacity for all 802.11ad $\pi/2$-QPSK modulated MCS indices; embedding conducted in first n parity bits versus last n data bits of each LDPC codeword.

Table 6.6. Estimated embedding coefficient, $\hat{r}_E$, for 802.11ad (last n Data Bits)

Modulation	**MCS**	**Rate** 1/2	**MCS**	**Rate** 5/8	**MCS**	**Rate** 3/4	**MCS**	**Rate** 13/16	**MCS**	**Rate** 7/8
$\pi/2$-BPSK	2	86.45	3	71.70	4	48.20	5	N/A	–	–
$\pi/2$-QPSK	6	81.66	7	67.01	8	48.42	9	46.41	9.1	13.33
$\pi/2$-16QAM	10	62.65	11	58.71	12	40.98	12.1	43.70	12.3	13.39
$\pi/2$-64QAM	–	–	12.3	47.15	12.4	37.31	12.5	38.28	12.6	14.94

MCS 5. The results from these trials were utilized to calculate the estimated embedding coefficients, $\hat{r}_E$, and are summarized in Table 6.6. When compared to the values for $\hat{r}_E$ obtained from the original embedding location, embedding in the last n data bits resulted in an increased embedding capacity for all MCS indices with the exception of MCS 12, and the MCS that utilize the punctured $R = 7/8$ LDPC code.

We also utilized the results of these extended trials to revisit the maximum embedding capacity, measured in bits-per-codeword, for all of the MCS indices. Based on a 4096-octet

Table 6.7. Maximum estimated embedding for 802.11ad DMG SC PHY, presented in bits-per-codeword; embedding conducted in first n parity bits and last n data bits.

MCS	**Modulation**	**Rate**	**Embedding Capacity per Codeword**	
			First *n* Parity Bits	Last *n* Data Bits
2	$\pi/2$-BPSK	$R = 1/2$	103 bits	111 bits
3		$R = 5/8$	87 bits	94 bits
4		$R = 3/4$	50 bits	54 bits
5		$R = 13/16$	N/A	N/A
6	$\pi/2$-QPSK	$R = 1/2$	95 bits	106 bits
7		$R = 5/8$	85 bits	91 bits
8		$R = 3/4$	48 bits	49 bits
9		$R = 13/16$	48 bits	53 bits
9.1		$R = 7/8$	21 bits	13 bits
10	$\pi/2$-16QAM	$R = 1/2$	96 bits	105 bits
11		$R = 5/8$	84 bits	91 bits
12		$R = 3/4$	49 bits	50 bits
12.1		$R = 13/16$	48 bits	52 bits
12.2		$R = 7/8$	23 bits	15 bits
12.3	$\pi/2$-64QAM	$R = 5/8$	84 bits	91 bits
12.4		$R = 3/4$	48 bits	50 bits
12.5		$R = 13/16$	47 bits	49 bits
12.6		$R = 7/8$	N/A	N/A

PSDU, the maximum embedding for each MCS are presented in Table 6.7; of note, there is no maximum embedding capacity listed for MCS 12.6 because our methodology could not establish a valid upper bound for this particular rate.

6.2 Convolutional Code Embedding

In this section, we outline significant results from our simulations conducted for embedding in convolutional codes. Once we established the behavior of our embedding techniques utilizing our MCS model, we transitioned to trials utilizing our 802.11ac testbed.

6.2.1 Variable-rate Embedding of Punctured Convolutional Codes

Initial simulations conducted with our BPSK MCS testbed validated the performance of the embedding techniques for MCS indices that utilized punctured code rates. Specifically, we identified that for MCS B, with a base $R = 2/3$ code, we were able to embed the coded bitstream at an interval of $\nu = 15$, producing a $R_{equiv} = 10/14$ embedded code with a BER performance that would fall between MCS B and MCS C. We repeated the same process for MCS C, with the embedding interval $\nu = 16$, or an $R_{equiv} = 12/15$ implementation. As shown in Figure 6.9, the fact that the BER performance of the embedded MCS falls between MCS rates I_C and I_{C+1} indicates that this technique would be valid in the case where the MCS was intentionally decremented.

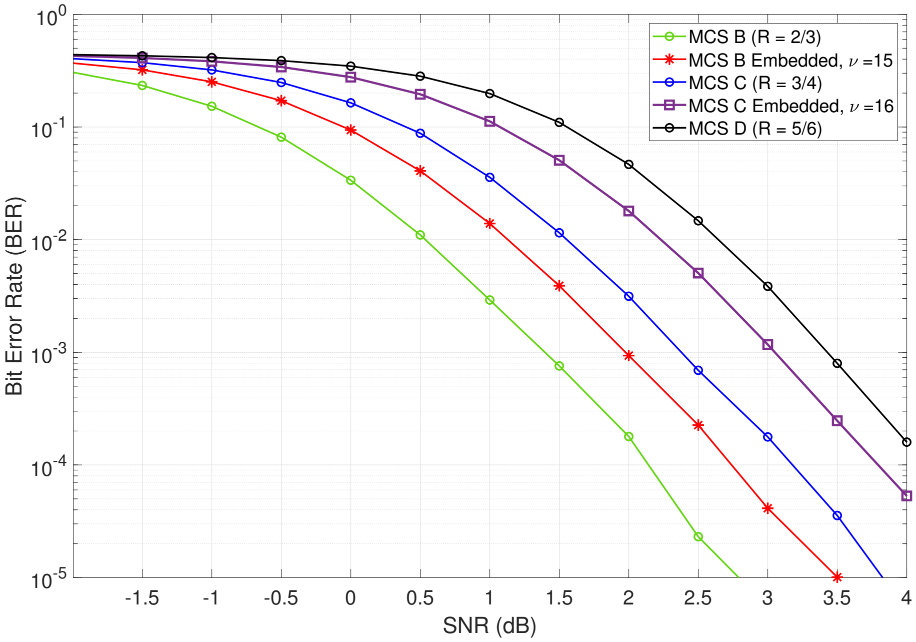


Figure 6.9. Bit error ratio versus SNR performance curves for punctured embedding implementation, base code rate $R = 2/3$, $\nu = 15$, and base code rate $R = 3/4$, $\nu = 16$. BPSK simulation over AWGN channel, 1200 bits per packet.

Following the successful demonstration of our initial scheme at the puncture position and ν values determined in our simulated trials, we looked to expand the flexibility of the embedding technique. We found that by increasing or decreasing the embedding interval, we had the ability to vary the embedding capacity of our channel. We were able to obtain increased embedding capacity while decreasing ν, at the expense of increasing the impact (measured in BER) on the underlying channel; conversely, increasing the interval ν reduced the SNR requirement while simultaneously decreasing embedding capacity. The results of our trials on MCS B, $R = 2/3$ embedding, can be seen in Figure 6.10 while the results for MCS C, $R = 3/4$ embedding, are shown in Figure 6.11.

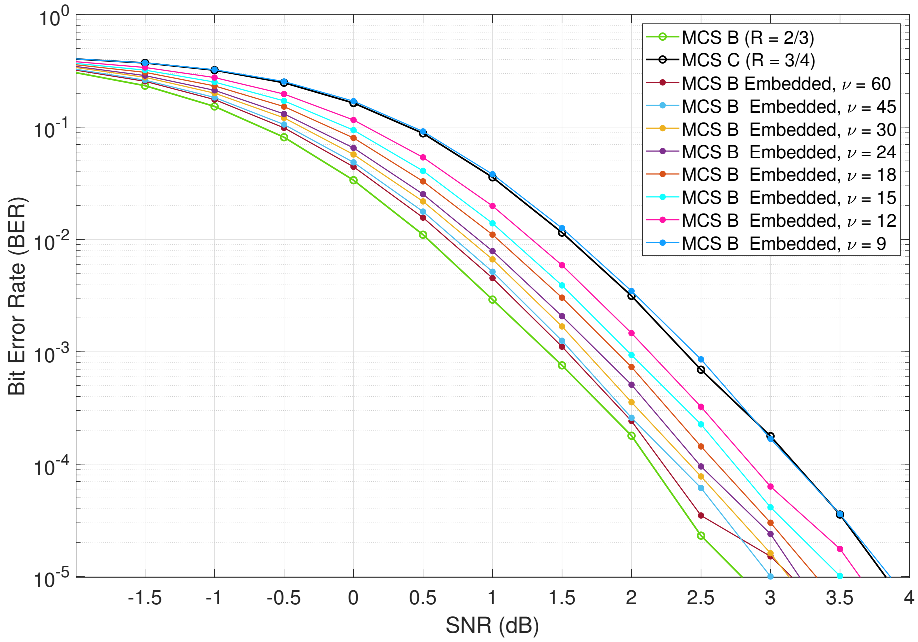


Figure 6.10. Bit error ratio versus SNR performance curves for variable rate embedding in MCS B; BPSK modulation, $R = 2/3$ code, AWGN channel.

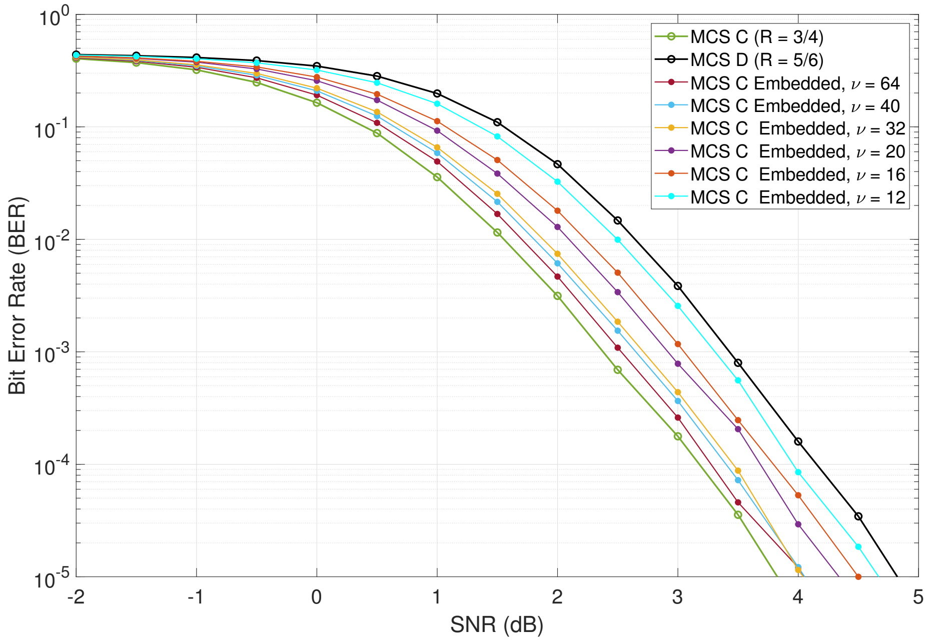


Figure 6.11. Bit error ratio versus SNR performance curves for variable rate embedding in MCS C; BPSK modulation, $R = 3/4$ code, AWGN channel.

For MCS B, we were able to successfully implement embedding intervals from $\nu = 9$ to 60 without exceeding the SNR requirements to support MCS C. MCS C embedding was likewise demonstrated for $\nu = 12$ to 64. By restricting the possible values of ν to multiples of n, where the base MCS code rate is $R = k/n$, we are able to maintain the same relative embedding position within the coded bit stream; this also allowed the continued use of the capacity equations defined in Section 5.4.

6.2.2 Embedding within IEEE 802.11ac VHT

Based on the results obtained from our simulations conducted within the BPSK-based embedding model, we extended the same techniques to our IEEE 802.11ac VHT testbed. Embedding trials were conducted for both unpunctured and punctured MCS over an AWGN channel. The MCS indices utilized for these simulations were configured with 8×8 MIMO, 8 spatial streams, and a BW of 80 MHz. Based on receiver sensitivity requirements established in [44], the PER threshold for a 4096-octet PSDU would be 10% vice the 1% for 802.11ad DMG.

Similar to our series of experiments for DMG, the trials conducted on our VHT testbed sought to demonstrate the feasibility of our embedding techniques for two use cases. The first was a situation where the MCS index was intentionally decremented to increase the redundancy and therefore embedding capacity of our channel. This case would be demonstrated through the embedding of the unpunctured parent code utilized by MCS 1; the goal was to facilitate the embedded channel while maintaining an acceptable PER assuming a channel state that could support MCS 2. The second case utilized variable rate embedding to exploit an embedding margin, M_E, within the existing MCS. To support this objective we would use the embedding technique discussed in Section 6.2.1.

Embedding Unpunctured VHT MCS

Despite the positive results obtained during the initial trials conducted in the simplified BPSK testbed, initial simulations of our unpunctured code embedding technique in 802.11ac VHT did not return a favorable outcome. As shown in Figure 6.12, despite implementation embedding and extraction in the same puncturing pattern used for the $R = 3/4$ code, our equivalent embedding technique returned substantially worse performance from a PER perspective.

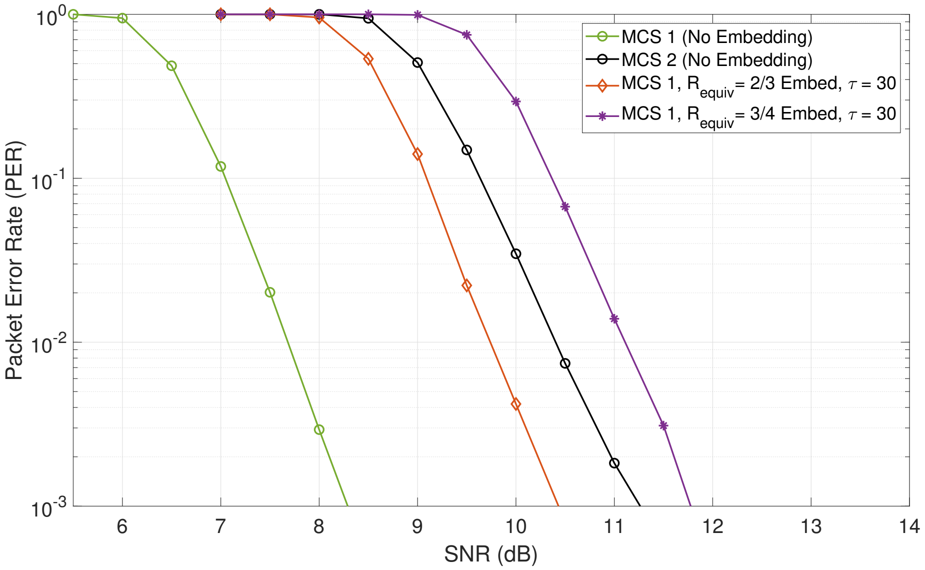


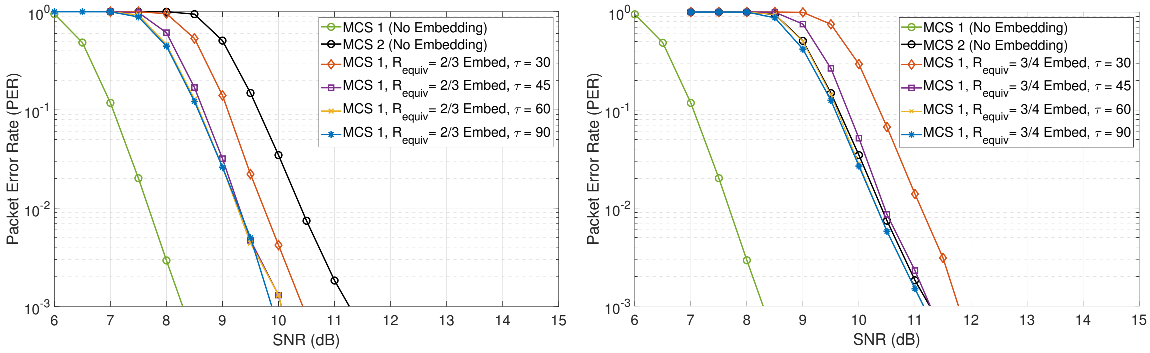
Figure 6.12. Unpunctured MCS embedding, VHT PHY simulation under AWGN channel. Results for MCS 1, 10000 PSDU per SNR, 4096-octet A-MPDU; decoder utilizing MCS 1 standard traceback depth, $\tau = 30$.

We quickly determined that this poor performance was due to the configuration of the Viterbi decoder and the selection of the traceback depth, τ. In the MATLAB implementation of the BCC decoder, $\tau = 30$ is utilized for the unpunctured $R = 1/2$ decoder utilized in MCS 1, while $\tau = 45$ is specified for $R = 2/3$ punctured codes and $\tau = 60$ is specified for the $R = 3/4$ code rate. This dependence on τ was not identified during our initial MCS model simulations because our configuration had utilized a set value of $\tau = 96$ for all MCS indices.

Recognizing that the performance of these codes, and therefore our embedding scheme, is highly dependent on the selection of τ, we conducted additional trials to characterize the impact of increased traceback depth. The results of those trials, which evaluated $\tau = 30$, 45, 60, and 90 are shown in Figure 6.13 for both $R_{equiv} = 2/3$ and $R_{equiv} = 3/4$ embedding.

In both cases, there was a noticeable performance gain based on the increased τ. As a result, the ability to adjust the traceback depth on the receiver would need to be investigated when evaluating this particular technique for implementation. The results of optimizing τ can be seen in Figure 6.14.

Assuming the ability to influence the selection of τ as part of our embedding implemen-



(a) (b)

Figure 6.13. PER performance comparison based on variations in τ for embedding trial conducted at MCS 1; 10000 trials per SNR, 4096-octet A-MPDU: (a) embedding at R_{equiv}=2/3 [ν_U =4, $b_{\nu,U}$ =1] and (b) embedding at R_{equiv}=3/4 [ν_U =6, $b_{\nu,U}$ =2].

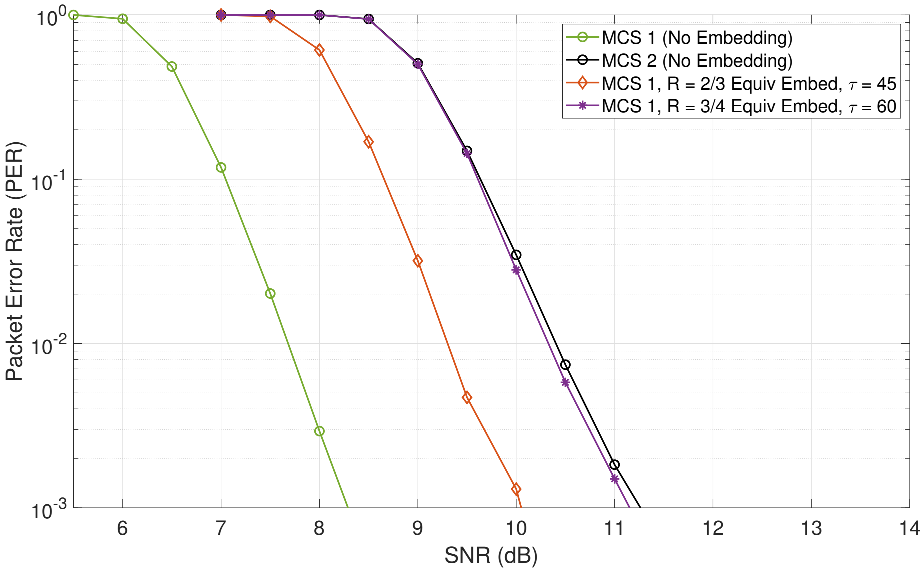


Figure 6.14. Unpunctured MCS embedding, VHT PHY simulation under AWGN channel. Results for MCS 1, 10000 trials per SNR, 4096-octet A-MPDU; decoder utilizing optimized traceback depth, τ, for equivalent embed rates.

tation, and utilizing (5.19) and (5.20), we are able to determine the size of the FEC-protected embedded payload, $L_{E,VHT}$, carried in each PPDU. The expected embedding capacity is summarized in Table 6.8. This table includes changes to the embedding capacity depending on various rates of FEC protection. When our embedding scheme is operating under a decremented MCS index, the capacity highlighted in grey is only available if we opt to utilize less error correction redundancy for the embedded payload than the underlying channel. For the case where MCS 1 is being utilized by the underlying communication system with a channel bandwidth of 80MHz, a 400ns guard interval, and $N_{SS} = 8$ spatial streams, the embedded data rate for $R_{Equiv} = 3/4$, including a FEC code of $R_{EC} = 1/2$, would be well in excess of 100 Mbps.

Table 6.8. Payload capacity of proposed embedding implementation of 802.11ac, MCS 1; data capacity in bits-per-PPDU under varying rates of error protection, R_{EC}. 8×8 MIMO, 8 spatial streams, 80-MHz BW, 4096-octet A-MPDU.

.

Selected FEC for Embedded Data	**Embedded Payload Capacity**	
	$\boldsymbol{R_{equiv} = 2/3}$ $[\nu_U = 4, b_{\nu,U} = 1]$	$\boldsymbol{R_{equiv} = 3/4}$ $[\nu_U = 6, b_{\nu,U} = 2]$
No FEC	16848 bits	22464 bits
$R_{EC} = 1/2$	8424 bits	11232 bits
$R_{EC} = 2/3$	11232 bits	14976 bits
$R_{EC} = 3/4$	12636 bits	16848 bits
$R_{EC} = 5/6$	14040 bits	18720 bits

Embedding Punctured VHT MCS

For already punctured MCS, embedding trials were conducted for MCS 5 and MCS 6 utilizing the embedding techniques previously discussed in Section 5.2.2. Unlike our initial VHT trials, the performance of our variable embedding in MCS 5 and 6 did not appear to suffer from the same issues involving insufficient traceback depth, τ. As with the proof-of-concept conducted in our BPSK MCS model, we selected a range of embedding intervals, ν, for each MCS implementation. For MCS 5, ν varied between 9 and 108; for MCS 6 it was between 12 and 96. The results of these embedding trials are shown in Figures 6.15 and 6.16 respectively.

Although the behavior of this embedding was very similar to our implementation in the MCS model, it is worthwhile to note that while it appeared that $\nu = 9$ was a viable embedding interval when it was implemented on MCS B, it clearly exceeded the SNR requirement of the next MCS index in our VHT MCS 5 embedding trial. This difference

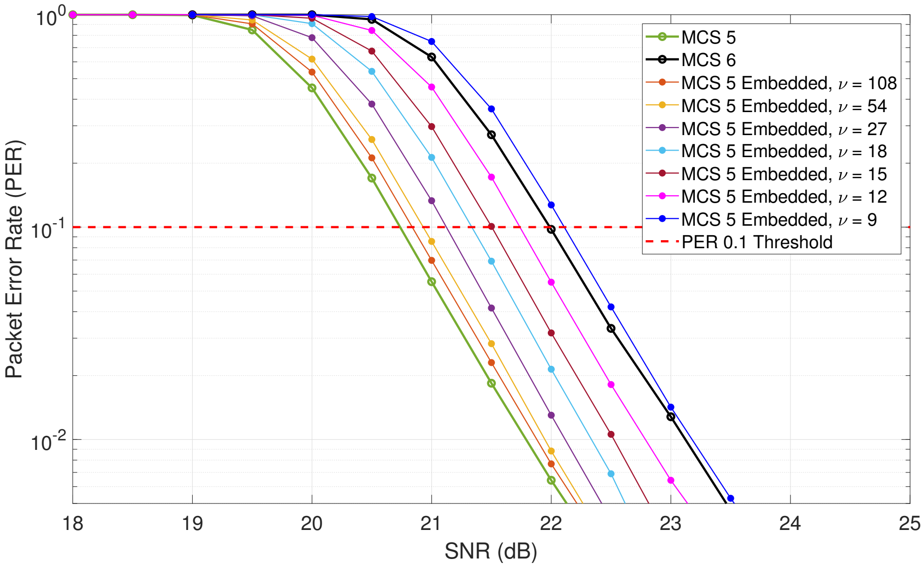


Figure 6.15. Variable rate embedding trials, VHT PHY simulation under AWGN channel. Results for MCS 5, 100000 trials per SNR, embedding capacity per 4096-octet A-MPDU.

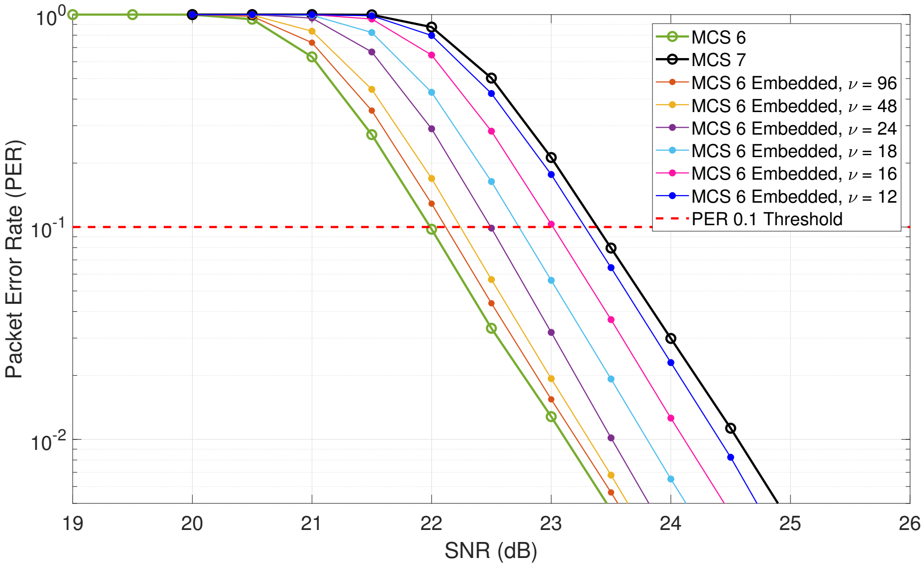


Figure 6.16. Variable rate embedding trials, VHT PHY simulation under AWGN channel. Results for MCS 6, 100000 trials per SNR, embedding capacity per 4096-octet A-MPDU.

in performance is likely due to the fact that BER is not necessarily analogous to PER in convolutional codes, and because the PSDU required in MCS 6 to transport a 4096-octet

A-MPDU is significantly shorter than the same frame in MCS 5. Based on the results of these trials, the embedding capacity for each MCS are provided in Tables 6.9 and 6.10. As with our results from MCS 1, the capacities are provided, in bits-per-PPDU, for varying rates of FEC redundancy.

Table 6.9. Payload capacity of proposed embedding implementation of 802.11ac, MCS 5; data capacity in bits-per-PPDU under varying rates of error protection, R_{EC}. 8×8 MIMO, 8 spatial streams, 80 MHz BW, 4096 octet A-MPDU.

.

Selected FEC for Embedded Data	**Embedded Payload Capacity (bits per-PPDU)**					
	$v = 12$	$v = 15$	$v = 18$	$v = 27$	$v = 54$	$v = 108$
No FEC	4680	3744	3120	2080	1040	520
$R_{EC} = 1/2$	2340	1872	1560	1040	520	260
$R_{EC} = 2/3$	3120	2496	2080	1386	693	346
$R_{EC} = 3/4$	3510	2808	2340	1560	780	390
$R_{EC} = 5/6$	3900	3120	2600	1733	866	433

Table 6.10. Payload capacity of proposed embedding implementation of 802.11ac, MCS 6; data capacity in bits-per-PPDU under varying rates of error protection, R_{EC}. 8×8 MIMO, 8 spatial streams, 80 MHz BW, 4096 octet A-MPDU.

.

Selected FEC for Embedded Data	**Embedded Payload Capacity (bits per-PPDU)**					
	$v = 12$	$v = 16$	$v = 18$	$v = 24$	$v = 48$	$v = 96$
No FEC	3744	2808	2496	1872	936	468
$R_{EC} = 1/2$	1872	1404	1248	936	468	234
$R_{EC} = 2/3$	2496	1872	1664	1248	693	312
$R_{EC} = 3/4$	2808	2106	1872	1404	624	351
$R_{EC} = 5/6$	3120	2340	2080	1560	780	390

6.3 Embedding under Multipath Fading Channels

The final series of experimental trials conducted during our investigation implemented multipath fading models to more accurately reflect real-world conditions. In both cases, the number of trials that could be conducted was limited based on the significant increase in computational resources required for each simulation. Even taking full advantage of performance gains offered by the MATLAB parallel processing toolbox, the time required to complete the simulations with multipath fading channels increased by an order of magnitude over those conducted in AWGN. As a result, the intent of presenting the following results is to not exhaustively describe the performance of these channels under the simulated fading conditions, but rather validate that the impact of these embedding techniques on the underlying channel were consistent with that observed under AWGN.

6.3.1 DMG Embedding with TGay Multipath Fading Channel

Our final set of LDPC embedding trials were conducted in a simulated multipath fading environment. We modified our DMG embedding testbed to incorporate the MATLAB TGay fading channel system object previously described in Section 4.7. We selected the SISO implementation of the open area model where the transmitters and receiver were separated by a distance of approximately 10 meters.

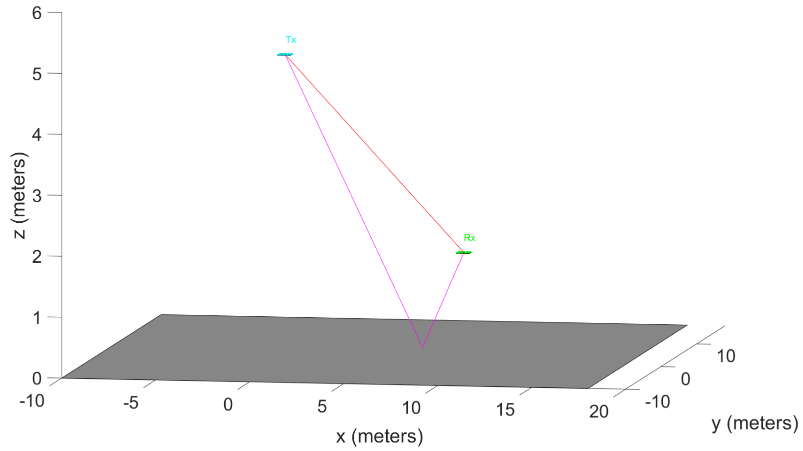


Figure 6.17. Open area hotspot model utilized to simulate 60-GHz DMG multipath fading environment

We also selected the number of elements for each antenna; despite the fact that 802.11ad does not support MIMO, it utilizes a uniform rectangular array (URA) antenna to support beamforming. We selected a 4 × 4 configuration based on available specification of real-world DMG implementations. The geometry of the resulting fading model is shown in Figure 6.17. With the channel model configured, we proceeded to repeat our embedding trials for MCS 6, testing various embedding rates from 1 bit-per-codeword up to a maximum of 120 bits-per-codeword. For each embedding rate, the simulation transmitted 100000 PSDU at every SNR point. The resulting PER performance curves, truncated after the maximum embedding capacity of 95 bits-per-codeword are shown in Figure 6.18.

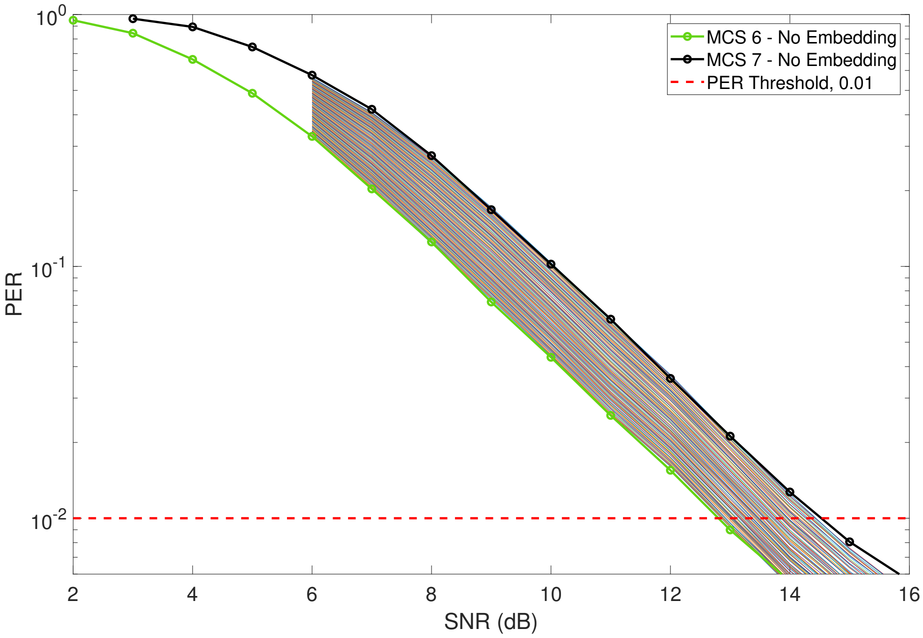


Figure 6.18. Packet error ratio versus SNR embedding trial, DMG PHY simulation over TGay multipath fading channel. Results for MCS 6, 100000 trials per SNR, 1 to 95 embedded bits per LDPC codeword.

Once we obtained the results of these trials we repeated the embedding coefficient analysis from Chapter 4 to determine if the previously identified relationship between the embedding margin, M_E, and the embedding capacity, C_{CW}, remained in the presence of multipath fading. The results of this analysis can be found in Figure 6.19; given the approximately linear trend of the embedding capacity, we were able to determine $\hat{r}_E = 56.9$.

Although this coefficient is substantially lower than that calculated from the AWGN case, its reduction is due entirely to the increased SNR between MCS 6 and 7; the maximum embedding rate per codeword remains 95 bits. Based on these results, we feel confident that the analytical techniques developed under AWGN would be applicable to fading environments.

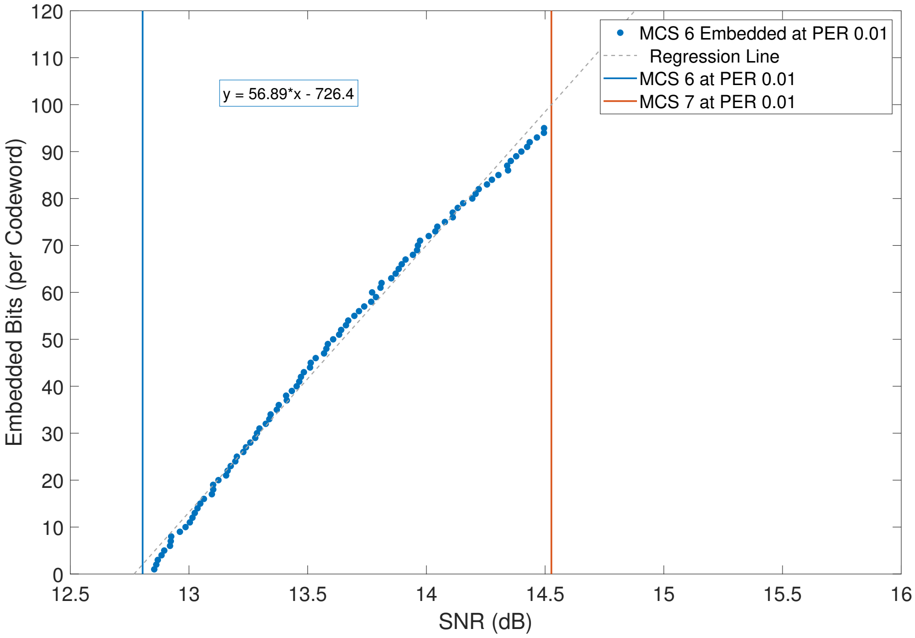


Figure 6.19. Estimated embedding capacity at a given SNR while maintaining 1% PER with associated line of regression. DMG PHY simulation over TGay multipath fading channel, MCS 6, 100000 PSDU per SNR point, 1 to 95 embedded bits per LDPC codeword.

6.3.2 VHT Embedding with TGac Multipath Fading Channel

Finally, we conducted a series of trials using the MATLAB TGac multipath fading model previously described in Section 5.6. The simulation was configured using the *Model-D* delay profile that best represents a typical office environment, and the transmission distance was set to 10 meters, which resulted in NLOS channel conditions.

We configured our original VHT testbed to filter our transmitted symbols through the desired multipath fading channel. Before conducting any embedding trials, we first needed to establish the performance of the VHT MCS under fading conditions; this performance was evaluated in terms of PER versus SNR. We again elected to utilize 8 spatial streams, 8×8 MIMO and a BW of 80 MHz. Once these initial performance bounds were determined, we completed embedding simulations for two of the cases that had been previously explored with an AWGN channel.

The first set of trials was conducted against a VHT MCS that utilized an unpunctured code rate. To align with our previous simulation, we repeated our embedding trials on

MCS 1 for $R_{equiv} = 2/3$ and $R_{equiv} = 3/4$. These trials were conducted using the optimal values for τ that were determined in Section 6.2.2. Based on our previous AWGN trials, we expected that the simulated PER of $R_{equiv} = 3/4$ with $\tau = 60$ would be consistent with the performance of the unembedded MCS 2. As shown in Figure 6.20, the performance of our embedding technique in the fading channel aligned with our expectations and supports the results obtained under AWGN conditions.

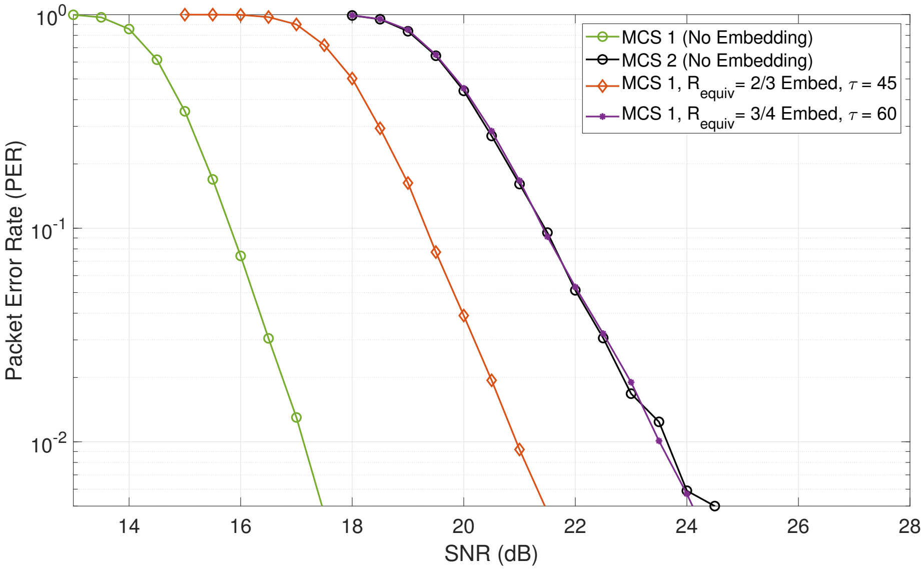


Figure 6.20. Unpunctured MCS embedding, VHT PHY simulation under TGac channel Model-D, 10m, NLOS. Results for MCS 1, 10000 trials per SNR, 4096-octet A-MPDU.

The second set of trials was conducted against VHT MCS 5, which utilizes the punctured $R = 2/3$ BCC. To replicate the original embedding trials, we repeated simulations for values of ν that ranged between 9 and 108. After the transmission of 10000 PPDU at each SNR point, we evaluated the PER performance of the underlying communications system; the results, plotted against the unembedded PER for VHT MCS 5 and 6, are shown in Figure 6.21.

The results obtained from this series of trials were also entirely consistent with those observed under AWGN. This outcome further supports our assertion that these error correction-based embedding techniques would be viable under real world conditions.

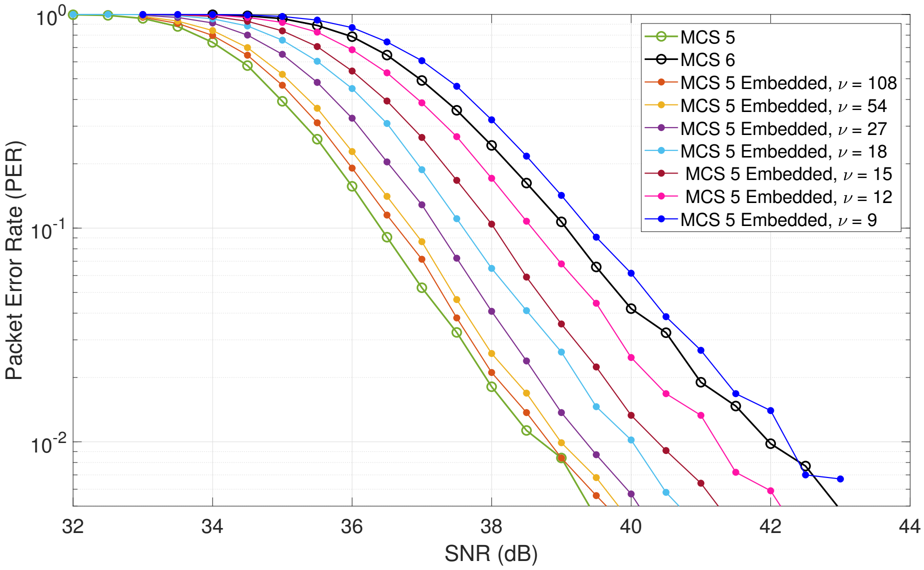


Figure 6.21. Variable rate embedding trials, VHT PHY simulation under TGac; Model-D, 10m, NLOS. Results for MCS 5, 10000 trials per SNR, embedding capacity per 4096-octet A-MPDU.

6.4 Summary

In this chapter, we presented the significant results of our MATLAB embedding simulations for both LDPC and convolutional codes. The simulated results support our initial observations on the ability to develop variable rate embedded channels within the error correction codes of modern WLAN standards. We further demonstrated the ability to control the amount of distortion experienced by the underlying channel by varying the rate of our embedding. Using the expressions developed in Chapters 4 and 5, we were able to evaluate the capacity of our embedding implementations. Finally, we extended our previous simulations to verify the performance of our embedding techniques in multipath fading channels.

CHAPTER 7: Conclusion

The goal of our work was to explore the implementation of embedded channels developed within error correction techniques utilized by modern wireless protocols. These methods exploit redundant error correction capacity within systems that can operate across a range of modulation and coding schemes; the resulting channels have the potential to support both covert communications as well as other legitimate auxiliary applications. We investigated specific methodologies based on the error correction mechanisms and PHY characteristics of two separate WLAN specifications. Finally, we developed models and analytical techniques to describe the behavior of our embedding schemes, provided capacity estimates for the resulting channels, and identified the impact to the performance of the underlying communication systems.

We accomplished the objectives of this dissertation through the development of embedding methodologies that have been demonstrated for both LDPC and convolutional codes and subsequently evaluated through simulation testbeds for IEEE 802.11ad DMG and 802.11ac VHT. These testbeds supported the extensive simulations utilized to characterize the behavior of the embedding schemes and, specifically, the interaction of our embedding technique to the MCS indices that enable rate adaptation in response to changes in channel conditions. These results enabled the development of analytical expressions to characterize the behavior of our techniques as well as establish performance bounds.

7.1 Significant Contributions

The research conducted in this dissertation has significantly advanced the body of work related to embedded channels within the error correction mechanisms of adaptive rate wireless communication systems. The first contribution was the development of an embedded channel model that greatly increased the available capacity by leveraging the inherent functionality of adaptive rate communication systems. The second contribution was the extension of our original implementation to support variable embedding rates and the development of analytical techniques to estimate the embedding capacity as it relates to current channel conditions. The final contribution was the development of constraints to the variable rate embedding that not only define the maximum embedding capacity, but also establish the minimum margin of embedding, M_E, required to meet our embedded channel requirements.

7.1.1 Embedding in Adaptive Rate Wireless Communication Systems

Extending on previous efforts that utilized error correction codes to develop covert communication channels [28], [33], [34], we identified that in adaptive rate communication systems, the effective redundancy of the error correction scheme could be increased by intentionally selecting a lower MCS index. Through the development of simulation testbeds, we were able to demonstrate the application of this concept to the original block code application in IEEE 802.11ad DMG while also extending these results to punctured convolutional codes that support IEEE 802.11ac and other legacy WLAN standards.

Beyond the basic proof-of-concept simulations, we were able to establish the capacity of this technique across multiple MCS indices as well as to examine the reliability of our embedded channel with respect to the error performance of the embedded payload. We explored the implications of implementing varying rates of FEC techniques to balance the needs of reliability against overall payload capacity.

7.1.2 Variable Rate Embedding and Capacity Estimation

Modern 802.11 specifications include provisions that enable stations to estimate the channel state to support both soft-decision demodulation and decoding as well as potentially influence rate adaptation [42], [43]. Applying this information to our original embedding scheme, we successfully demonstrated that our embedding techniques could be tuned to support varied embedding capacities, which had differing impacts on the performance of the underlying channel. Based on extensive simulated trials, we were able to establish that in order to maintain a specified error threshold for the underlying channel, a relationship existed between the quality of the channel and the embedding capacity.

We also explored the concept of an embedding coefficient, r_E, which provided an estimate for the embedding capacity based on a specified margin of embedding, M_E, or the difference in SNR between the current channel conditions and the minimum requirement for a given MCS index. While our specific implementation in the IEEE 802.11ad standard resulted in a constant value of r_E for a given MCS, r_E is a function of the selected error correction code as well as the specific embedding technique and location. Even if this combination does not result in a constant value over the entire embedding range, this concept could be extended to factor in the available embedding margin and still provide reliable capacity estimates.

7.1.3 Embedding Model with Constraints

We developed a model for which we could specify reasonable limits on the available SNR, the maximum acceptable PER of the underlying communication system, and the minimum capacity requirements of our embedded channel. We recognized that if our proposed embedding methodologies were applied to real-world systems, there would be several internal and external factors that would limit the performance of our embedding scheme; we specifically focused on how the available embedding region, which is bounded by adjacent MCS indices, would be influenced by these various constraints. By incorporating these factors into our previous expressions for embedding capacity, we were able to improve our estimates for the maximum embedding payload as well as establish a threshold for the minimum required channel state to meet mission requirements.

7.2 Future Work

The results presented in this dissertation provide a basis for the analysis and performance assessment of embedded channels within adaptive rate wireless communication systems. While we spent significant time outlining the analysis of available capacity contained within these MCS-based adaptive rate communication systems, we would need to address two significant challenges before real-world implementation, particularly if the embedding was intended to support the development of a covert channel. Additionally, there are several areas related to the proposed embedding methodology that could be extended to improve our understanding of these embedding techniques or improve the embedding capacity of these schemes.

7.2.1 Real-World Implementation Challenges

The first challenge related to real-world implementation of these techniques is how to coordinate the embedding location and embedding rate between the transmitter and receiver.

While the location selected for embedding could be passed as a pre-shared key, the specific embedding rate would be more difficult to coordinate in a covert manner. Due to the low-data rate requirement for passing rate selection announcements, it might be possible to utilize an alternate path for this control channel such as embedding in an unused protocol field or in padded bit locations. An alternate solution might be to maintain an extremely low-rate embedding regardless of channel condition. This low capacity channel, which might only occupy a single bit per codeword, could act as a preamble for the full-rate channel and provide notification to the receiver of the location and size of the embedded payload before transmission.

The second issue involves having sufficient insight into the current channel state to make embedding decisions based on the available channel margin. As noted in Chapter 2, there has been significant research into SNR-based link adaptation algorithms that rely on accurate channel estimation information; it would be necessary to expand on this research to determine whether sufficiently accurate representations of the embedding margin exist to implement our proposed variable-rate embedding scheme.

7.2.2 Extensions to Proposed Methodology

Although we performed simulated trials of all potential embedding locations using the implementation described for the binary convolutional code, we did not perform an exhaustive search of the LDPC codes utilized within 802.11ad. There are no published puncturing patterns proposed for the specific LDPC codes utilized in DMG, but a possible extension of this work would be to utilize one of the puncturing algorithms or computer search methods to determine the ideal embedding locations; we could then repeat our simulations to determine if this optimized embedding resulted in increased capacity.

Our focus here was developing techniques that would maximize the capacity of the embedding channel. To that end, we proposed extracting and replacing the embedded bit locations to minimize the distortion to the underlying communications channel at these extremely high embedding rates. While this technique maximized capacity, it might require extensive hardware and firmware modifications and precludes the ability to develop a covert channel to a non-cooperative third party receiver. A simple extension of these proposed schemes would be to evaluate the embedding capacity for a technique that did not rely on the replacement of the received embedded values with LLR = 0; the resulting channel would trade capacity for simplicity of implementation.

While we concentrated our efforts on the implementation of these embedding techniques on IEEE 802.11 protocols, it should be possible to extend the application to other MCS-based adaptive rate wireless communication systems, including LTE. Although the error correction codes utilized to support these implementations may differ from those explored in our work, the use of distinct MCS indices, along with the implicit and explicit information related to the current channel state provided by these protocols, should enable the development of similar methodologies to support embedded channels.

Error correction code-based information hiding techniques have been traditionally limited to implementation in the PHY. Consequently, the embedded channels proposed in our research only span a single hop. A possible extension of these techniques could focus on implementation within application layer-FEC (AL-FEC); these codes have been developed for use within the application or transport layer to support streaming media with the goal of improving the reliability of content delivery [85]. In addition to AL-FEC enabling the development of end-to-end embedded channels, the fact that these codes are designed for use at the application layer means that they are intended to be implemented in software [86], which drastically reduces the potential complexity needed to exploit and modify hardware or firmware as required for our current proposed implementation.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX: Code Repository

The code contained in this appendix is representative of the MATLAB functions and scripts that were utilized to support our research into the development of embedded channels in adaptive rate wireless communication systems.

A.1 Low-Density Parity Check Code Embedding

The code contained in this section was developed to explore the structure of the LDPC codes utilized by IEEE 802.11ad DMG and also evaluate the performance of our embedding scheme in terms of the metrics PER and BER.

A.1.1 Construction of LDPC Parity Check Matrices

The following script was developed to investigate the structure of the QC-LDPC parity check matrices specified for IEEE 802.11ad. The parity check matrices were constructed based on [44].

```
%% LDPCParityMatricesDMG
%  This script was developed to create the LDPC parity check matrices specified
%  for IEEE 802.11ac.  Included function of cyclicPermMatrix, which construct the
%  appropriate shifted identity matrix based on the table entry specified in IEEE
%  802.11-2016.
format compact;
clear all;
% Table entries to define parity check matrices as defined in 802.11ad Standard;
% parity table entries for LDPC rate-1/2 code
ParityTable = [40,-1,38,-1,13,-1,5,-1,18,-1,-1,-1,-1,-1,-1,-1;
                   34,-1,35,-1,27,-1,-1,30,2,1,-1,-1,-1,-1,-1,-1;
                   -1,36,-1,31,-1,7,-1,34,-1,10,41,-1,-1,-1,-1,-1;
                   -1,27,-1,18,-1,12,20,-1,-1,-1,15,6,-1,-1,-1,-1;
                   35,-1,41,-1,40,-1,39,-1,28,-1,-1,3,28,-1,-1,-1;
                   29,-1,0,-1,-1,22,-1,4,-1,28,-1,27,-1,23,-1,-1;
                   -1,31,-1,23,-1,21,-1,20,-1,-1,12,-1,-1,0,13,-1;
                   -1,22,-1,34,31,-1,14,-1,4,-1,-1,-1,13,-1,22,24];
% Z represents the size of the permutation matrix defined for 802.11ad standard
Z = 42;
% Determine total size of parity check matrix based on parity table and
% permutation matrix
ParityTableSize = size(ParityTable);
```

```
ParityTableRow = ParityTableSize(1,1);
ParityTableCol = ParityTableSize(1,2);
% Fill Matrix will all zeros
LDPCParity = zeros(ParityTableSize*Z);
% Develop parity matrix from provided table and permutation matrix
for row = 1:ParityTableRow;
    rowLower = ((row-1)*Z)+1;
    rowUpper = row*Z;
    for col = 1:ParityTableCol;
        tableEntry = ParityTable(row,col);
        MatrixElement = cyclicPermMatrix(Z,tableEntry);
        colLower = ((col-1)*Z)+1;
        colUpper = col*Z;
        LDPCParity(rowLower:rowUpper,colLower:colUpper)=MatrixElement;
    end
end
% Save resulting matrix in Excel format
filename = 'testdata.xlsx';
xlswrite(filename,LDPCParity);

%% Supporting function cyclicPermMatrix
function [matrixElement] = cyclicPermMatrix(Z,tableEntry)
% Develops appropriate shifted version of identity matrix
    if tableEntry < 0;
        % Creates Z x Z matrix of zeros
        matrixElement = zeros(Z);
    else
        % Creates Z x Z identity matrix
        P = eye(Z);
        % Circular Shift; opposite of of tableEntry based on published standard
        matrixElement = circshift(P,[-tableEntry,0]);
    end
end
```

A.1.2 Simulation for IEEE 802.11ad DMG Embedding

The original simulation for IEEE 802.11ad was developed from a PER calculation example provided as part of the release notes for the WLAN Toolbox in MATLAB 2018a. Although the specific script we used to develop our main simulation is no longer available online, an updated version was published with MATLAB release 2018b; this version also contained support for TGay multipath fading [87].

In order to implement the simulation it was necessary to modify both the base script as well as some of the supporting WLAN Toolbox functions. The primary functions that required modification were `wlan.internal.dmgData`, which encodes the underlying PSDU before transmission, and `wlanDMGDataBitRecover` which demodulates and decodes the received PSDU. The relationship of these MATLAB Toolbox functions is shown in Figure A.1.

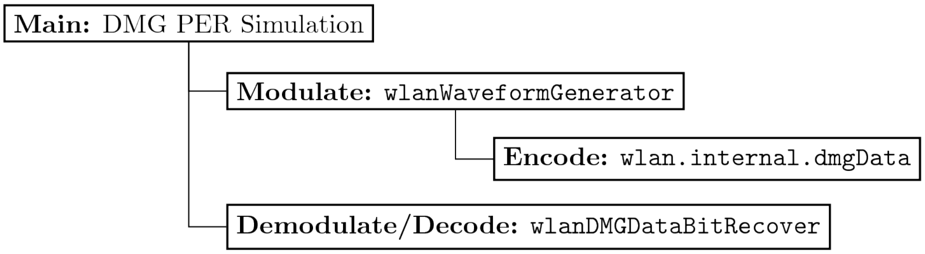

Main: DMG PER Simulation
Modulate: wlanWaveformGenerator
Encode: wlan.internal.dmgData
Demodulate/Decode: wlanDMGDataBitRecover

Figure A.1. Structure of PER simulation for 802.11ad-based embedding utilizing MATLAB WLAN Toolbox

On the transmit side, the `wlanWaveformGenerator` function must also be modified to allow for the additional parameters needed for embedding to be passed to the updated version of `wlan.internal.dmgData`.

Main Simulation, DMG PER

From the main program, embedding is accomplished by calling the modified version of the `wlanWaveformGenerator` function. The desired payload is then forwarded to a modified version of `wlan.internal.dmgData` and embedded into the LDPC codewords. The below code segment also applies FEC-protection to the embedded payload.

```
%% Encode embed data with appropriate FEC
% Determine maximum number of embedded bits per codeword based on
% simulation parameters
embedBitsPerCW = embedBitsIndex(embedTrial);
% Select Data to Embed (array format)
embedDataArray = messageArray(1:embedDataSize(embedTrial));
% Reshape embed data into codeword
embedBlkCW = reshape(embedDataArray,embedCWdataLen,numEmbedCW(embedTrial));
% Encode Parity Bits (Embed Data)
embedParityBits = wlan.internal.ldpcEncodeCore(embedBlkCW,mcsEmbedTable.Rate);
% Check if selected FEC is rate-7/8 (requires puncturing)
if isequal(mcsEmbedTable.Rate,7/8)
    % Puncture first 48 parity bits if needed
    embedParityBitsPunc = embedParityBits(49:end,:);
    ldpcEmbedBits = [embedBlkCW; embedParityBitsPunc];
```

```
else
    ldpcEmbedBits = [embedBlkCW;embedParityBits];
end
% Reshape embedded data into column
embedData = reshape(ldpcEmbedBits,[],1);
%% wlanWaveformGenertor_embedDataFEC
%  Calls modified version of 'wlanWaveformGenerator'.  Passes the additional
%  arguments, 'embedData' and 'embedBitsPerCW', necessary to enable
%  embedding in the LDPC codewords; embedding occurs in the function
%  'wlan.internal.dmgData'.
txWaveform = wlanWaveformGenerator_embedDataFEC(psdu,cfgDMG,embedData,embedBitsPerCW);
```

Extraction of the FEC-protected embedded payload is accomplished through modification of the `wlanDMGDataBitRecover` function. This section of code also performs the LDPC decode for the FEC-protected embedded payload, determines the number of bit errors (and BER) found in the embedded payload, and calculates the number of packet errors encountered in the underlying communications channel. These values are later used to determine the PER and BER for the trials conducted at each SNR point.

```
%% Embedded payload extraction and zero stuffing
%  Calls modified version of 'wlanDMGDataBitRecover'.  Function returns the soft
%  decisions values of the underlying communication channel as well as the LLR
%  values associated with the embedded bit positions.
[dataDecode,softEmbedBits] = wlanDMGDataBitRecover_embedDataFEC(rxDataSym,nVarEst,...
embedBitsPerCW,cfgEmbedDMG,cfgDMG);
% Reshape embed data into LDPC codewords
softEmbedBitsCW = reshape(softEmbedBits,paramsEmbedMCS.LCW,...
    numEmbedCW(embedTrial));
% Section of code required to perform LDPC decoding on the embedded data
maxLDPCIterationCount = 12;
if isequal(mcsEmbedTable.Rate,7/8)
    % Add punctured 48 parity bits and decode with 13/16 rate
    softEmbedBitsCWPunc =...
    [softEmbedBitsCW(1:546,:);zeros(48,numEmbedCW(embedTrial));...
        softEmbedBitsCW(546+1:end,:)];
    [decodedEmbedBits,numIterations,parityCheck] =...
    wlan.internal.ldpcDecodeCore(softEmbedBitsCWPunc,13/16,maxLDPCIterationCount);
else
    [decodedEmbedBits,numIterations,parityCheck] =...
    wlan.internal.ldpcDecodeCore(softEmbedBitsCW,mcsEmbedTable.Rate,...
    maxLDPCIterationCount);
end
%% Hard decision decode (no FEC) is also performed on returned LLR values
% Returned codewords truncated to remove parity bits
softEmbedBitsData = softEmbedBitsCW(1:embedCWdataLen,:);
```

```
% Hard decision, LLR > 0 interpreted as a binary '0'
softEmbedBitsData(softEmbedBitsData>0)=0;
% Hard decision, LLR < 0 interpreted as a binary '1'
softEmbedBitsData(softEmbedBitsData<0)=1;
rcvEmbedBitsNoFEC = softEmbedBitsData;

% Determine if there was a packet error in the underlying communication channel
packetError = any(biterr(psdu,dataDecode));
% Compute total number of packet errors (cumulative over all trials)
numPacketErrors = numPacketErrors+packetError;
% Identity bit errors (and BER) for the embedded payload
dataRcv = dataDecode;
% Number of bit errors (and BER) when FEC is utilized
[embedErrorsTrial,embedBERTrial] = biterr(embedBlkCW,decodedEmbedBits);
% Number of bit errors (and BER) for hard decision decode, no FEC
[embedErrorsTrialNoFEC,embedBERTrialNoFEC] = biterr(embedBlkCW,rcvEmbedBitsNoFEC);
% Compute total number of bit errors (over all trials) when FEC is utilized
numEmbedErrors = numEmbedErrors + embedErrorsTrial;
% Compute total number of bit errors (over all trials), no FEC
numEmbedErrorsNoFEC = numEmbedErrorsNoFEC + embedErrorsTrialNoFEC;
```

Modifications to WLAN Toolbox Function for Payload Embedding

The following modified version of the MATLAB function `wlan.internal.dmgData` was utilized to enable embedding of our desired payload. This particular implementation was for embedding the interleaved FEC-protected payload in the first *n* parity bits of each LDPC codeword.

```
1   function y = dmgData_embedDataFEC(psdu,cfgDMG,embedData,embedBitsPerCW)
2   %% dmgData DMG Data field processing of the PSDU
3   %  Copyright 2016-2017 The MathWorks, Inc.
4
5   %% Embedding Modifications: dmgData_embedDataFEC
6   %  The original wlan.internal.dmgData function developed by MATLAB for the WLAN
7   %  Toolbox has been modified to conduct error correction code-based embedding
8   %  within IEEE 802.11ad DMG; this version implements interleaved FEC-protected
9   %  embedding and replaces the call for 'wlan.internal.dmgData' in the
10  %  original MATLAB-developed simulation.  All header comments from the
11  %  original function have been removed; lines 14 - 25 are unmodified from original
12  %  'wlan.internal.dmgData' function.  Function modifications occur between the
13  %  dashed lines.
14
15      if strcmp(phyType(cfgDMG),'Control')
16          % Encode header and data together due to differential encoding
17          headerBits = wlan.internal.dmgHeaderBits(cfgDMG);
18          encHeaderBits = wlan.internal.dmgHeaderEncode(headerBits,psdu,cfgDMG);
```

```
        encDataBits = wlan.internal.dmgDataEncode(psdu,cfgDMG);
        % Modulate
        yT = wlan.internal.dmgDataModulate([encHeaderBits; encDataBits],cfgDMG);
        % Strip out the encoded data from differential modulation
        y = yT((8192+1):end);
    else % SC/OFDM PHY
        encodedStream = wlan.internal.dmgDataEncode(psdu,cfgDMG);
    % -------------------------------------------------------------------------------
        %% Embed Data (FEC Protected, with Interleaving)
        % Gather parameters based on selected MCS
        paramsMCS = wlan.internal.dmgSCEncodingInfo(cfgDMG);
        mcsTable = wlan.internal.getRateTable(cfgDMG);
        % Determine padding added during encode process; retain padding
        encodedStreamPadSize = paramsMCS.NBLK_PAD;
        encodeStreamPad = encodedStream((end-(encodedStreamPadSize-1):end));
        % Length of LDPC codewords within encoded stream
        encodedStreamCW = numel(encodedStream) - encodedStreamPadSize;
        % Reshape encoded data stream into LDPC Codewords; padding removed
        encodedCW = reshape(encodedStream(1:encodedStreamCW),paramsMCS.LCW,[]);
        % Data Bits in each CW
        dataBitsPerCW = paramsMCS.LCW * mcsTable.Rate;
        % Calculates start and stop position for embedding (first n bits of data CW)
        embedStartLoc = dataBitsPerCW + 1;
        embedEndLoc = embedStartLoc + embedBitsPerCW - 1;
        % Defines area of CW available for embedding based on first n bits of data CW
        embedArea = encodedCW(embedStartLoc:embedEndLoc,:);
        % Size of Embed Area
        embedAreaSize = size(embedArea);
        % Reshape embedArea into Array (Row-wise Reshape)
        embedArray = reshape(embedArea.',[],1);
        embedArrayWithData = embedArray;
        % Embed data into desginated positions
        embedArrayWithData(1:numel(embedData)) = embedData;
        % Reshape embedArrayWithData into embedArea (Row-wise)
        embedAreaWithData = ...
        reshape(embedArrayWithData.',embedAreaSize(2),embedAreaSize(1)).';
        encodedCWplusEmbed = encodedCW;
        encodedCWplusEmbed(embedStartLoc:embedEndLoc,:) = embedAreaWithData;
        % Reshape matrix into single bitstream
        encodedBits = [reshape(encodedCWplusEmbed,[],1);encodeStreamPad];
    % -------------------------------------------------------------------------------
        % Modulate Embedded Bitstream
        y = wlan.internal.dmgDataModulate(encodedBits,cfgDMG);
    end

end
```

Modifications to WLAN Toolbox Function for Payload Extraction and Recovery

Recovery of the embedded data, as well as replacement of the embedded locations with LLR = 0, is accomplished with a modified version of the MATLAB WLAN Toolbox function `wlanDMGDataBitRecover`.

```
1   function [dataBits,softEmbedBits] = wlanDMGDataBitRecover_embedDataFEC(rx,...
2   noiseVarEst,embedBitsPerCW,cfgEmbedDMG,varargin)
3   %% wlanDMGDataBitRecover Recover data bits from DMG Data field
4   %  Copyright 2017 The MathWorks, Inc.
5
6   %% Embedding Modifications: wlanDMGDataBitRecover_embedDataFEC
7   %  This function has been modified to extract embedded bit locations within
8   %  IEEE 802.11ad DMG and stuff those bit locations with LLR = 0; this version
9   %  implements extraction of interleaved FEC-protected embedded payload and
10  %  replaces the call for 'wlanDMGDataBitRecover' in the original simulation. All
11  %  header comments from the original function have been removed; lines 14 - 77 are
12  %  unmodified from original 'wlandDMGDataBitRecover' function.  Function
13  %  modifications occur between the dashed lines.
14
15  % Check minimum and maximum number of input arguments
16      narginchk(5,10)%(3,8);
17      % If input rx is empty then do not attempt to decode; return empty
18      if isempty(rx)
19          dataBits = zeros(0,1,'int8');
20          return;
21      end
22      csiFlag = 0;
23      if isa(varargin{1},'wlanDMGConfig')
24          % If no CSI input is present
25          cfgDMG = varargin{1};
26          csi = [];
27      elseif nargin>5 && isa(varargin{2},'wlanDMGConfig')
28          %elseif nargin>3 && isa(varargin{2},'wlanDMGConfig')
29          csi = varargin{1};
30          cfgDMG = varargin{2};
31          csiFlag = 1;
32      else
33          coder.internal.error('wlan:shared:ExpectedDMGObject');
34      end
35      % Validate configuration object
36      validateattributes(cfgDMG,{'wlanDMGConfig'},{'scalar'},mfilename,'DMG format ...
            configuration object');
37      % Input CSI is only required for OFDM PHY
38      coder.internal.errorIf(¬isempty(csi) && ¬...
            strcmp(phyType(cfgDMG),'OFDM'),'wlan:shared:InvalidInputCSI');
39      % Validate each P-V pair
40      if isempty(coder.target) % Simulation path
```

```
        p = inputParser;
        p.PartialMatching = true;
        % Set defaults for the optional arguments
        addParameter(p,'MaximumLDPCIterationCount',12);
        addParameter(p,'EarlyTermination',false);
        parse(p,varargin{2+csiFlag:end}); % Parse inputs
        res = p.Results;
        maximumLDPCIterationCount = res.MaximumLDPCIterationCount;
        earlyTermination = res.EarlyTermination;
    else % Codegen path
        pvPairs = struct('MaximumLDPCIterationCount', uint32(0), ...
                     'EarlyTermination', uint32(0));
        % Select parsing options
        popts = struct('PartialMatching', true);
        % Parse inputs
        pStruct = ...
            coder.internal.parseParameterInputs(pvPairs,popts,varargin{2+csiFlag:end});
        % Get values for the P-V pair or set defaults for the optional arguments
        maximumLDPCIterationCount = ...
        coder.internal.getParameterValue(pStruct.MaximumLDPCIterationCount,12,...
        varargin{2+csiFlag:end});
        earlyTermination = ...
        coder.internal.getParameterValue(pStruct.EarlyTermination,false,...
        varargin{2+csiFlag:end});
    end
    validateattributes(maximumLDPCIterationCount,{'numeric'},{'real','nonempty',...
    'scalar','finite','>',0},mfilename,'''MaximumLDPCIterationCount''value');
    validateattributes(earlyTermination,{'logical'},...
    {'scalar','nonempty'},mfilename,'''EarlyTermination'' value');
    % Validate input
    validateattributes(rx,{'double'},{'2d','finite'},mfilename,'input');
    % Validate input noise estimate
    validateattributes(noiseVarEst,{'double'},...
    {'real','scalar','nonnegative','finite'},mfilename,'noiseVarEst');
    if csiFlag
        softBits = wlan.internal.dmgDataDemap(rx,noiseVarEst,csi,cfgDMG);
    else
        softBits = wlan.internal.dmgDataDemap(rx,noiseVarEst,cfgDMG);
    end
% --------------------------------------------------------------------------------
    %% Embedded Data Recovery and Zero Stuffing
    % Gather characteristics about selected MCS
    paramsMCS = wlan.internal.dmgSCEncodingInfo(cfgDMG);
    mcsTable = wlan.internal.getRateTable(cfgDMG);
    % Determine number of total demodulated bits
    softBitsSize = size(softBits);
    % Reshape into column array
    softBitsArray = reshape(softBits,[],1);
    % Determine padding added during encode process
    softBitsPadSize = paramsMCS.NBLK_PAD;
    softBitsPad = softBitsArray((end-(softBitsPadSize-1):end));
    % Length of LDPC codewords within encoded stream
    softBitsCWLen = numel(softBits) - softBitsPadSize;
    % Reshape encoded data stream into LDPC codewords; padding removed
    softBitsCW = reshape(softBits(1:softBitsCWLen),paramsMCS.LCW,[]);
    % Calculate data Bbts in each CW
    dataBitsPerCW = paramsMCS.LCW * mcsTable.Rate;
    % Determine embedding location within codewods; only valid for embedding ...
        within first n parity bits
    embedStartLoc = dataBitsPerCW + 1;
    embedEndLoc = embedStartLoc + embedBitsPerCW - 1;
    embedArea = softBitsCW(embedStartLoc:embedEndLoc,:);
    % Size of Embed Area
    embedAreaSize = size(embedArea);
    % Reshape embedArea into Array (Row-wise)
    embedArray = reshape(embedArea.',[],1);
    % Determine number of Embedded Data Bits in Embed Area
    paramsEmbedMCS = wlan.internal.dmgSCEncodingInfo(cfgEmbedDMG);
    mcsEmbedTable = wlan.internal.getRateTable(cfgEmbedDMG);
    % Number of embedded data CW per packet
    numEmbedCW = floor((embedBitsPerCW * paramsMCS.NCW)/paramsEmbedMCS.LCW);
    %Embed Data CW Size
    embedCWLen = paramsEmbedMCS.LCW;
    % Determine total size of Embeded CW
    embedDataSize = numEmbedCW*embedCWLen;
    % Select embedded bit locations; second argument returned from function
    softEmbedBits = embedArray(1:embedDataSize);
    embedArrayNoEmbed = embedArray;
    % Stuff LLR = 0 values into embedded locations
    embedArrayNoEmbed(1:embedDataSize) = 0;
    % Reshape embedArrayWithData into embedArea (Row-wise)
    embedAreaNoEmbed = ...
        reshape(embedArrayNoEmbed.',embedAreaSize(2),embedAreaSize(1)).';
    softBitsNoEmbed = softBitsCW;
    softBitsNoEmbed(embedStartLoc:embedEndLoc,:) = embedAreaNoEmbed;
    % Reshape embedded bits into original dimensions
    softBitsStream = [reshape(softBitsNoEmbed,[],1);softBitsPad];
% ---------------------------------------------------------------------------------
    %% Decode Legitimate Data; first argument returned from function
    dataBits = wlan.internal.dmgDataDecode(softBitsStream,cfgDMG,...
    maximumLDPCIterationCount,earlyTermination);
end
```

A.1.3 Specify Changes in Embedding Location

Our original simulations embedded the payload in the first *n* parity bits of the LDPC codewords. A modified technique enabled the selection of alternate embedding locations.

This section briefly describes the code modifications required to enable this functionality; of note, this particular implementation did not FEC-protect the embedded payload.

```
%% The code provided below specifies changes that must be made to the main script
%  of the PER simulation in order to support user-specified changes to the
%  embedding location:

% The embed location must be defined.
embedLocation = "begPar";
% This parameter will then be passed via an updated call to
% 'wlanWaveformGenerator_embedData' to embed the data; as with the original
% implementation, this function passes the arguments through to a modified
% version of the 'wlan.internal.dmgData' function, in this case
% called 'dmgData_embedData'.
txWaveform = wlanWaveformGenerator_embedData(psdu,cfgDMG,embedData,embedLocation);
% The embedLocation parameter is also passed to the
%  'wlanDMGDataBitRecover_embedData' function to extract the embedded bits and
% replace them with LLR = 0.
[dataDecode,rcvEmbedBits] = wlanDMGDataBitRecover_embedData(rxDataSym,...
                nVarEst,embedBitsPerCW,cfgDMG,embedLocation);
```

Modifications were also required to the original `wlan.internal.dmgData` function in order to support the user-specified embedding locations. The segment of code below would replace the code modifications to `wlan.internal.dmgData` previously identified in Appendix A.1.2.

```
%% Embedding Modifications: dmgData_embedData for multiple embedding locations.
%  This code replaces the modifications identified in dmgData_embedDataFEC.
% -------------------------------------------------------------------------------
    %% Embed Data (No FEC, user-defined location)
    % Gather parameters based on selected MCS
    paramsMCS = wlan.internal.dmgSCEncodingInfo(cfgDMG);
    mcsTable = wlan.internal.getRateTable(cfgDMG);

    encodedCW = reshape(encodedStream(1:size(encodedStream)-...
    paramsMCS.NBLK_PAD),paramsMCS.LCW,[]);
    %% location of embed data
    embedBitsPerCW = size(embedData,1)/(paramsMCS.NCW);
    dataBitsPerCW = paramsMCS.LCW * mcsTable.Rate;
    % Calculates start and stop bits for embedding based on selected location
    if embedLocation == "begData"  % Embed first n data bits
        embedStartLoc = 1;
        embedEndLoc = embedStartLoc + embedBitsPerCW - 1;
    elseif embedLocation == "midData"  % Embed middle n data bits
```

```
        embedStartLoc = dataBitsPerCW/2 + 1;
        embedEndLoc = embedStartLoc + embedBitsPerCW - 1;
    elseif embedLocation == "endData"  % Embed last n data bits
        embedStartLoc = dataBitsPerCW - embedBitsPerCW + 1;
        embedEndLoc = dataBitsPerCW;
    elseif embedLocation == "begPar"  % Embed first n parity bits
        embedStartLoc = dataBitsPerCW + 1;
        embedEndLoc = embedStartLoc + embedBitsPerCW - 1;
    elseif embedLocation == "midPar"  % Embed middle n parity bits
        % subtracts total bits from data, to get amount of parity bits,
        % dividing it by two to get the midpoint
        parityMidVal = (paramsMCS.LCW - dataBitsPerCW)/2;
        embedStartLoc = paramsMCS.LCW - parityMidVal + 1;
        embedEndLoc = embedStartLoc + embedBitsPerCW - 1;
    elseif embedLocation == "endPar"  % Embed last n parity bits
        embedStartLoc = paramsMCS.LCW - embedBitsPerCW + 1;
        embedEndLoc = paramsMCS.LCW;
    end
    % Reshape embedded payload into appropriate dimensions
    embedDataCW = reshape(embedData,embedBitsPerCW,[]);
    encodedCWplusEmbed = encodedCW;
    % Embed payload into locations determined above
    encodedCWplusEmbed(embedStartLoc:embedEndLoc,:) = embedDataCW;
    encodedBits = [reshape(encodedCWplusEmbed,[],1);encodedStream(...
    (end-(paramsMCS.NBLK_PAD-1):end))];
% --------------------------------------------------------------------------------
```

User-specified embedding locations also required updates to the function `wlanDMGDataBitRecover`. The below modifications would replace those previously identified for this function in Appendix A.1.2.

```
%% Embedding Modifications: wlanDMGDataBitRecover_embedData for multiple
%% embedding locations.
%  This code replaces the modifications identified in %  ...
  wlanDMGDataBitRecover_embedDataFEC.
% --------------------------------------------------------------------------------
    %% Embedded Data Recovery and Zero Stuffing
    % Gather characteristics about selected MCS
    paramsMCS = wlan.internal.dmgSCEncodingInfo(cfgDMG);
    mcsTable = wlan.internal.getRateTable(cfgDMG);
    % Reshape into column array
    softBitsArray = reshape(softBits,[],1);
    softBitsCW = ...
        reshape(softBits(1:size(softBitsArray,1)-paramsMCS.NBLK_PAD),paramsMCS.LCW,[]);
    %% location of embed data
    % embedBitsPerCW has already been passed to this function and is available
    dataBitsPerCW = paramsMCS.LCW * mcsTable.Rate; % How many data bits are in the ...
```

```
            codeword
        % Calculates start and stop bits of payload based on selected location
        if embedLocation == "begData"  % Payload in first n data bits
            embedStartLoc = 1;
            embedEndLoc = embedStartLoc + embedBitsPerCW - 1;
        elseif embedLocation == "midData"  % Payload in middle n data bits
            embedStartLoc = dataBitsPerCW/2 - embedBitsPerCW +1;
            embedEndLoc = embedStartLoc + embedBitsPerCW - 1;
        elseif embedLocation == "endData"  % Payload in last n data bits
            embedStartLoc = dataBitsPerCW - embedBitsPerCW + 1;
            embedEndLoc = dataBitsPerCW;
        elseif embedLocation == "begPar"  % Payload in first n parity bits
            embedStartLoc = dataBitsPerCW + 1;
            embedEndLoc = embedStartLoc + embedBitsPerCW - 1;
        elseif embedLocation == "midPar"  % Payload in middle n parity bits
            parityMidVal = (paramsMCS.LCW - dataBitsPerCW)/2;
            % subtracts total bits from data, to get amount of parity bits,
            % dividing it by two to get the midpoint
            embedStartLoc = paramsMCS.LCW - parityMidVal + 1;
            embedEndLoc = embedStartLoc + embedBitsPerCW - 1;
        elseif embedLocation == "endPar"  % Payload in last n parity bits
            embedStartLoc = paramsMCS.LCW - embedBitsPerCW + 1;
            embedEndLoc = paramsMCS.LCW;
        end
        % Conduct hard decision decoding on embedded bits (no FEC applied)
        hardEmbedBits = softBitsCW(embedStartLoc:embedEndLoc,:);
        hardEmbedBits(softEmbedBits>0)=0;
        hardEmbedBits(softEmbedBits<0)=1;
        rcvEmbedBits = reshape(hardEmbedBits,[],1);
        softBitsNoEmbed = softBitsCW;
        % Stuff LLR = 0 values into embedded locations
        softBitsNoEmbed(embedStartLoc:embedEndLoc,:) = 0;
        % Reshape received bits (with embedding locations removed) prior to decoder
        softBitsStream = ...
            [reshape(softBitsNoEmbed,[],1);softBitsArray(end-(paramsMCS.NBLK_PAD-1):end)];
    % -------------------------------------------------------------------------------
```

A.2 Variable Rate Embedding Analysis Tools

In order to fully characterize the performance of our embedding scheme, it became necessary to export the embedding trial results, retain them for future analysis, and develop a semi-logarithmic interpolation tool to estimate the required SNR for each given embedding rate.

A.2.1 Data Collection from PER Trials

The following code was appended to our existing PER testbed to save the vital metrics from each simulation in a uniquely-named .mat file.

```
%% Data Collection and Structure Array Development
%  The following code section were appended to our embedding simulations to record
%  relevant metrics related to the embedding performance, specifically, the PER
%  of the underlying system and BER of embedded payload.

% Generate a structure array with fields that correspond to the relevant
% performance metrics from each trial
trialResults.embedBitsIndex = embedBitsIndex; % Embedded bits per codeword
trialResults.SNRvalues = SNRvalues; % SNR points evaluated for simulation
trialResults.packetErrorRate = packetErrorRate; %PER
trialResults.embedBER = embedBER; %BER
% Establish DTG format for timestamp
formatOut = 'yyyymmdd_HHMMSS';
% Generate filename for .mat file that contains the results.
% Unique name based on simulation characteristics and timestamp
fileName = ['MCS_',num2str(mcsFilenameIndex(mcsValue)),'_MCSEmbed_',...
    num2str(mcsFilenameIndex(mcsValue+1)),'_PER_and_BER_',...
    num2str(trialResults.embedBitsIndex(1)),'-',...
    num2str(trialResults.embedBitsIndex(end)),'_',num2str(maxNumErrors),'-',...
    num2str(maxNumPackets),'_',num2str(datestr(now,formatOut)),'_',...
    num2str(embedLocation)];
% Save resulting output
save(fileName, 'trialResults');
fprintf('Results saved to %s\n', fileName);
```

A.2.2 Semi-logarithmic Interpolation and Plotting

This code determined the estimated SNR required to maintain a specified PER at given embedding rate by performing semi-logarithmic interpolation.

```
%% Estimate SNR to maintain specified PER threshold
%  Script must be in the same path as the files that need to be combined.  Prior to
%  running this script, the user should load the .mat file containing the first
%  dataset into the MATLAB Workspace.

figure;
% Plot all PER vs SNR curves in dataset
semilogy(trialResults.SNRvalues,trialResults.packetErrorRate,'HandleVisibility','off')
hold on
% Set PER threshold based on protocol specification or user preference
```

```
threshold = 0.01;
% Perform semi-log interpolation on each dataset
for numEmbedBits = 1:length(trialResults.embedBitsIndex);
    aboveThreshold = find(trialResults.packetErrorRate(numEmbedBits,:) ≥ threshold);
    upperBoundLoc(numEmbedBits) = aboveThreshold(end);
    if upperBoundLoc(numEmbedBits) < length(trialResults.SNRvalues);
        belowThreshold = find(trialResults.packetErrorRate(numEmbedBits,:) ≤ ...
            threshold);
        lowerBoundLoc(numEmbedBits) = belowThreshold(1);
        if upperBoundLoc(numEmbedBits) == lowerBoundLoc(numEmbedBits);
            SNRatThreshold(numEmbedBits) = ...
                trialResults.SNRvalues(lowerBoundLoc(numEmbedBits));
        else
            upperBoundPER(numEmbedBits) = ...
                trialResults.packetErrorRate(numEmbedBits,upperBoundLoc(numEmbedBits));
            upperBoundSNR(numEmbedBits) = ...
                trialResults.SNRvalues(upperBoundLoc(numEmbedBits));
            lowerBoundPER(numEmbedBits) = ...
                trialResults.packetErrorRate(numEmbedBits,lowerBoundLoc(numEmbedBits));
            lowerBoundSNR(numEmbedBits) = ...
                trialResults.SNRvalues(lowerBoundLoc(numEmbedBits));
            SNRatThreshold(numEmbedBits)= ...
                upperBoundSNR(numEmbedBits)+((log10(threshold)- ...
                log10(upperBoundPER(numEmbedBits)))/ ...
                (log10(lowerBoundPER(numEmbedBits))- ...
                log10(upperBoundPER(numEmbedBits))) ...
                \*(lowerBoundSNR(numEmbedBits)- upperBoundSNR(numEmbedBits));
        end
    end
end
% Plot estimated SNRpoint against the PER curves
plot(SNRatThreshold,threshold,'r+','HandleVisibility','off')

% Plot Embedding Rate vs SNR
figure;
plot(SNRatThreshold,trialResults.embedBitsIndex(1:length(SNRatThreshold)),'+')
hold on
clear all;
```

A.3 Convolutional Code Embedding

We created a simple software simulation to conduct our initial investigation of convolutional embedding. This code was based on a MATLAB-developed simulation that compared the performance of hard and soft decision Viterbi decoders under QAM [88].

```matlab
%% Adaptive Rate Embedding Model
%  Software testbed to establish BER performance of binary convolutional code
%  and evaluate embedding locations using equivalent puncture patterns for
%  the specified code rates.

clear all;

M = 2;                  % Modulation order
b = log2(M);            % Bits per symbol
EbNoVec = (0:0.5:8)';   % Eb/No values (dB)
numSymPerFrame = 1200;  % Number of PSK symbols per frame
numTrials = 10;         % Number of trials

% Specify characteristics of the desired convolutional code
constraintLen = 7;      % Constrain Length (K)
codeG1 = 133;           % First generator polynomial
codeG2 = 171;           % Second generator polynomial
trellis = poly2trellis(constraintLen,[codeG1 codeG2]);
tbd = 96;               % Traceback depth
% Specify Code Rate
k = 2;                  % Numerator of code rate
n = 3;                  % Denominator of code rate
rate = k/n;             % Specified code rate

% Initialize array for BER results
BERSoft = zeros(numTrials,length(EbNoVec));
% Specify puncture patterns based on the selected code rate
if rate == 1/2;         % Rate for MCS 'A'
    puncpat = [1 1];
elseif rate == 2/3;     % Rate for MCS 'B'
    puncpat = [1 1 1 0];
elseif rate == 3/4;     % Rate for MCS 'C'
    puncpat = [1 1 1 0 0 1];
elseif rate == 5/6;     % Rate for MCS 'D'
    puncpat = [1 1 1 0 0 1 1 0 0 1];
elseif rate == 10/14;   % Req = 10/14 (from 2/3)
    puncpat = [1 0 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0];
else                    % Req = 12/15 (from 2/3)
    puncpat = [1 0 1 0 0 1 1 1 1 0 0 1 1 1 1 0 0 1 1 1 1 0 0 1];
end

for trial = 1:numTrials;
    disp(trial)
    parfor num = 1:length(EbNoVec) % Parallel toolbox, for debug use 'for'
        % Convert Eb/No to SNR
        snrdB = EbNoVec(num) + 10*log10(b*rate);
        % Noise variance calculation for unity average signal power.
        noiseVar = 10.^(-snrdB/10);
        % Reset the error and bit counters
        [numErrsSoft,numErrsHard,numBits] = deal(0);
        while numErrsSoft < 100 && numBits < 1e8  %Specify stopping parameters
            % Generate binary data
            dataIn = randi([0 1],numSymPerFrame*b,1);
            % Convolutionally encode the data
            dataEnc = convenc(dataIn,trellis,puncpat);
            % PSK modulate
            H = comm.PSKModulator('ModulationOrder',M,'PhaseOffset',0,...
                'BitInput',true);
            txSig = step(H,dataEnc);
            % Pass through AWGN channel
            rxSig = awgn(txSig,snrdB,'measured');
            % PSK demodulate
            decision = 'Approximate log-likelihood ratio';
            I = comm.PSKDemodulator('ModulationOrder',M,'PhaseOffset',0,...
                'BitOutput',true,'DecisionMethod',decision,'VarianceSource',...
                'Property','Variance',noiseVar);
            rxDataSoft = step(I,rxSig);
            % Viterbi decode the demodulated data
            dataSoft = vitdec(rxDataSoft,trellis,tbd,'cont','unquant',puncpat);
            % Determine number of received errors
            numErrsInFrameSoft = biterr(dataIn(1:end-tbd),dataSoft(tbd+1:end));
            % Increment the error and bit counters
            numErrsSoft = numErrsSoft + numErrsInFrameSoft;
            numBits = numBits + numSymPerFrame*b;
        end
        % Calculate BER for trial
        BERSoft(trial,num) = numErrsSoft/numBits;
    end
end

% Determine average BER across the total number of specified trials
avgBERSoft = mean(BERSoft,1);

% Specify selected code rate
div = "%d/%d";
codeRate = sprintf(div,k,n);
% Record results from relevant infromation about trail in trialResults
% array; data collected to allow future analysis
trialResults.modOrder = M;
trialResults.codeRate = codeRate;
trialResults.puncPattern = puncpat;
trialResults.EbNovalues = EbNoVec;
trialResults.berEstSoft = BERSoft;

%% Save Trial Results
% Set DTG format to differentiate runs
formatOut = 'yyyymmdd_HHMMSS';
% Define file name
filenameDefault = ['Conv Code_K=',num2str(constraintLen),'_[',...
num2str(codeG1),'-',num2str(codeG2),']_tbd_',num2str(tbd),'_Rate_k=',...
num2str(k),'_n=',num2str(n),'_PSK_ModOrder_',...
num2str(trialResults.modOrder),'_SymPerFrame_',num2str(numSymPerFrame),...
'_numTrials_',num2str(numTrials),'_',num2str(datestr(now,formatOut))];
% Save run characteristics to .mat file in execution directory
fileName = filenameDefault;
save(fileName, 'trialResults');
fprintf('Results saved to %s\n', fileName);

%% Plot Results, BER vs Eb/No
semilogy(EbNoVec,avgBERSoft,'-*');
legend(sprintf('R = %d/%d',k,n));
grid;
xlabel('Eb/No (dB)');
ylabel('Bit Error Rate');
```

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

[1] W. Mazurczyk, S. Wendzel, S. Zander, A. Houmansadr, and K. Szczypiorski, *Information Hiding in Communication Networks: Fundamentals, Mechanisms, Applications, and Countermeasures (IEEE Press Series on Information and Communication Networks Security)*. Wiley-IEEE Press, 2016.

[2] J. Lubacz, W. Mazurczyk, and K. Szczypiorski, "Principles and overview of network steganography," *IEEE Communications Magazine*, vol. 52, no. 5, pp. 225–229, May 2014.

[3] T. G. Handel and M. T. Sandford, "Hiding data in the OSI network model," in *Information Hiding*, vol. 1174, G. Goos, J. Hartmanis, J. Leeuwen, and R. Anderson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 23–38. Available: http://link.springer.com/10.1007/3-540-61996-8_29

[4] "Cisco Visual Networking Index: Forecast and Methodology, 2016–2021," https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html, accessed: 2018-05-28.

[5] P. M. B. Harley, M. Tummala, and J. C. McEachen, "High-throughput covert channels in adaptive rate wireless communication systems," in *2019 International Conference on Electronics, Information, and Communication (ICEIC)*, Jan. 2019, pp. 1–7.

[6] K. Cabaj, L. Caviglione, W. Mazurczyk, S. Wendzel, A. Woodward, and S. Zander, "The new threats of information hiding: The road ahead," *IEEE IT Prof.*, vol. 20, no. 3, pp. 31–39, May 2018.

[7] W. Mazurczyk and L. Caviglione, "Information hiding as a challenge for malware detection," *IEEE Security Privacy*, vol. 13, no. 2, pp. 89–93, Mar. 2015.

[8] L. Caviglione, S. Wendzel, and W. Mazurczyk, "The future of digital forensics: Challenges and the road ahead," *IEEE Security Privacy*, vol. 15, no. 6, pp. 12–17, Nov. 2017.

[9] L. Frikha, Z. Trabelsi, and W. El-Hajj, "Implementation of a covert channel in the 802.11 header," in *2008 Int. Wireless Commun. and Mobile Computing Conference*, Aug. 2008, pp. 594–599.

[10] S. Cabuk, C. E. Brodley, and C. Shields, “IP covert timing channels: Design and detection,” in *Proceedings of the 11th ACM Conference on Computer and Communications Security* (CCS ’04). New York, NY, USA: ACM, 2004, pp. 178–187. Available: http://doi.acm.org/10.1145/1030083.1030108

[11] R. Holloway and R. Beyah, “Covert DCF: A DCF-based covert timing channel in 802.11 networks,” in *2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems*, Oct 2011, pp. 570–579.

[12] M. M. Sadek, A. S. Khalifa, and M. G. Mostafa, “Video steganography: A comprehensive review,” *Multimedia Tools Appl.*, vol. 74, no. 17, pp. 7063–7094, Sep. 2015. Available: http://dx.doi.org/10.1007/s11042-014-1952-z

[13] T. Rabie and M. Baziyad, “The pixogram: Addressing high payload demands for video steganography,” *IEEE Access*, vol. 7, pp. 21 948–21 962, 2019.

[14] R. Patel and M. Patel, “Steganography over video file by hiding video in another video file, random byte hiding and LSB technique,” in *2014 IEEE International Conference on Computational Intelligence and Computing Research*, Dec 2014, pp. 1–5.

[15] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, “Techniques for data hiding,” *IBM Systems Journal*, vol. 35, no. 3.4, pp. 313–336, 1996.

[16] E. Jones, O. L. Moigne, and J. Robert, “IP traceback solutions based on time to live covert channel,” in *Proceedings. 2004 12th IEEE International Conference on Networks (ICON 2004) (IEEE Cat. No.04EX955)*, 2004, vol. 2, pp. 451–457 vol.2.

[17] U. Tupakula, V. Varadharajan, and S. K. Vuppala, “Counteracting DDoS attacks in WLAN,” in *Proceedings of the 4th International Conference on Security of Information and Networks* (SIN ’11). ACM, 2011, pp. 119–126. Available: http://doi.acm.org/10.1145/2070425.2070445

[18] R. Clayton, S. J. Murdoch, and R. N. M. Watson, “Ignoring the great firewall of China,” in *Privacy Enhancing Technologies*. Springer Berlin Heidelberg, 2006, pp. 20–35.

[19] S. Aryan, H. Aryan, and J. A. Halderman, “Internet censorship in Iran: A first look,” in *Presented as part of the 3rd USENIX Workshop on Free and Open Communications on the Internet*. Washington, D.C.: USENIX, 2013. Available: https://www.usenix.org/conference/foci13/internet-censorship-iran-first-look

[20] W. Mazurczyk and Z. Kotulski, “New security and control protocol for VoIP based on steganography and digital watermarking,” *Annales UMCS Informatica*, vol. 5, pp. 417–426, 2006.

[21] D. Kahn, “The history of steganography,” in *Information Hiding* (Lecture Notes in Computer Science). Springer, Berlin, Heidelberg, May 1996, pp. 1–5.

[22] B. Carrara and C. Adams, “A survey and taxonomy aimed at the detection and measurement of covert channels,” in *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security* (IH&MMSec ’16). New York, NY, USA: ACM, 2016, pp. 115–126. Available: http://doi.acm.org/10.1145/2909827.2930800

[23] J. Fridrich, *Steganography in Digital Media: Principles, Algorithms, and Applications*. Cambridge University Press, 2010.

[24] C. Krätzer, J. Dittmann, A. Lang, and T. Kühne, “WLAN steganography: A first practical review,” in *Proceedings of the 8th Workshop on Multimedia and Security* (MM&Sec ’06). New York, NY, USA: ACM, 2006, pp. 17–22. Available: http://doi.acm.org/10.1145/1161366.1161371

[25] S. Zander, G. Armitage, and P. Branch, “A survey of covert channels and countermeasures in computer network protocols,” *IEEE Communications Surveys Tutorials*, vol. 9, no. 3, pp. 44–57, Third 2007.

[26] H. Zhao, “Covert channels in 802.11e wireless networks,” in *2014 Wireless Telecommunications Symp.*, Apr. 2014, pp. 1–5.

[27] J. Classen, M. Schulz, and M. Hollick, “Practical covert channels for WiFi systems,” in *2015 IEEE Conference on Communications and Network Security (CNS)*, Sep. 2015, pp. 209–217.

[28] I. Grabska and K. Szczypiorski, “Steganography in WiMAX networks,” in *2013 5th Int. Congr. Ultra Modern Telecommun. and Control Syst. and Workshops (ICUMT)*, Sep. 2013, pp. 20–27.

[29] S. Grabski and K. Szczypiorski, “Steganography in OFDM symbols of fast IEEE 802.11n networks,” in *2013 IEEE Security and Privacy Workshops*, May 2013, pp. 158–164.

[30] J. Fridrich and D. Soukal, “Matrix embedding for large payloads,” *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 3, pp. 390–395, Sep 2006.

[31] T. Filler and J. J. Fridrich, “Binary quantization using belief propagation with decimation over factor graphs of LDGM codes,” *CoRR*, vol. abs/0710.0192, 2007. Available: http://arxiv.org/abs/0710.0192

[32] A. Gautam and R. Gupta, “Enhancement of steganography scheme based on QC-LDPC codes,” in *2015 International Conference on Signal Processing and Communication (ICSC)*. IEEE, Mar 2015.

[33] X. Yan, S. Guan, and X. Niu, “Research on the capacity of error-correcting codes-based information hiding,” in *2008 International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 2008, pp. 1158–1161.

[34] K. S. Subramani, A. Antonopoulos, A. A. Abotabl, A. Nosratinia, and Y. Makris, “INFECT: INconspicuous FEC-based Trojan: A hardware attack on an 802.11a/g wireless network,” in *2017 IEEE Int. Symp. Hardware Oriented Security and Trust (HOST)*, May 2017, pp. 90–94.

[35] P. N. Safier, I. S. Moskowitz, and P. Cotae, “On the baseband communication performance of physical layer steganography,” in *2011 45th Annual Conference on Information Sciences and Systems*, 2011, pp. 1–6.

[36] E. Perahia and R. Stacey, *Next Generation Wireless LANs*. Cambridge University Pr., 2013.

[37] D. Hui, S. Sandberg, Y. Blankenship, M. Andersson, and L. Grosjean, “Channel coding in 5G New Radio: A tutorial overview and performance comparison with 4G LTE,” *IEEE Vehicular Technology Magazine*, vol. 13, no. 4, pp. 60–69, Dec 2018.

[38] S. Biaz and S. Wu, “Rate adaptation algorithms for IEEE 802.11 networks: A survey and comparison,” in *2008 IEEE Symposium on Computers and Communications*. Marrakech: IEEE, July 2008, pp. 130–136.

[39] M. Vutukuru, H. Balakrishnan, and K. Jamieson, “Cross-layer wireless bit rate adaptation,” in *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication* (SIGCOMM ’09). New York, NY, USA: ACM, 2009, pp. 3–14.

[40] M. Souryal and N. Moayeri, “Joint rate adaptation and channel-adaptive relaying in 802.11 ad hoc networks,” in *MILCOM 2006*. Washington, DC, USA: IEEE, Oct. 2006, pp. 1–8.

[41] I. Haratcherev, K. Langendoen, R. Lagendijk, and H. Sips, “Hybrid rate control for IEEE 802.11,” in *Proceedings of the Second International Workshop on Mobility Management &Amp; Wireless Access Protocols* (MobiWac ’04). New York, NY, USA: ACM, 2004, pp. 10–18. Available: http://doi.acm.org/10.1145/1023783.1023787

[42] J. Zhang, K. Tan, J. Zhao, H. Wu, and Y. Zhang, “A practical SNR-guided rate adaptation,” in *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*, Apr. 2008, pp. 2083–2091.

[43] X. Chen, P. Gangwal, and D. Qiao, “Practical rate adaptation in mobile environments,” in *2009 IEEE International Conference on Pervasive Computing and Communications*, March 2009, pp. 1–10.

[44] "IEEE standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pp. 1–3534, Dec. 2016.

[45] M. Lei and Y. Huang, "CFR and SNR estimation based on complementary Golay sequences for single-carrier block transmission in 60-GHz WPAN," in *2009 IEEE Wireless Communications and Networking Conference*, April 2009, pp. 1–5.

[46] B. Schulz, "White Paper: 802.11ad - WLAN at 60 GHz - Solution," Rohde & Schwarz GmbH & Co KG, Tech. Rep., 2017. Available: https://www.rohde-schwarz.com/us/solutions/wireless-communications/wlan-wifi/in-focus/white-paper-802-11ad-wlan-at-60-ghz_229148.html

[47] T. Nitsche, C. Cordeiro, A. B. Flores, E. W. Knightly, E. Perahia, and J. C. Widmer, "IEEE 802.11ad: Directional 60 GHz communication for multi-Gigabit-per-second Wi-Fi [Invited Paper]," *IEEE Commun. Mag.*, vol. 52, no. 12, pp. 132–141, Dec. 2014.

[48] P. Bright, "Intel to stop making WiGig cards for laptops but still pushing 60GHz for VR [Updated]," Sep. 2017. Available: https://arstechnica.com/gadgets/2017/09/intel-to-stop-making-wigig-cards-for-laptops-but-still-pushing-60ghz-for-vr/

[49] P. Kumari, N. Gonzalez-Prelcic, and R. W. Heath, "Investigating the IEEE 802.11ad standard for millimeter wave automotive radar," in *2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall)*, Sep. 2015, pp. 1–5.

[50] B. Su, "The next generation wireless LAN standard and overcome the test challenges," https://www.keysight.com/upload/cmc_upload/All/20170608-A3-BrianSu.pdf, Keysight, Tech. Rep., June 2017.

[51] T. S. Rappaport, R. W. Heath, R. C. Daniels, and J. N. Murdock, *Millimeter Wave Wireless Communications*. Pearson Education (US), 2014.

[52] C. Hemanth and T. G. Venkatesh, "Performance analysis of service periods (SP) of the IEEE 802.11ad hybrid MAC protocol," *IEEE Transactions on Mobile Computing*, vol. 15, no. 5, pp. 1224–1236, May 2016.

[53] T. K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*. John Wiley & Sons Inc, 2005.

[54] Z. Tu and S. Zhang, "Overview of LDPC codes," in *7th IEEE International Conference on Computer and Information Technology (CIT 2007)*. IEEE, Oct 2007.

[55] H. Ibl and C. Klaus, "White Paper: DOCSIS 3.1 - Application Note," Rohde & Schwarz GmbH & Co KG, Tech. Rep., 2015. Available: https://www.rohde-schwarz.com/us/applications/docsis-3.1-white-paper_230854-108554.html

[56] T. Richardson and S. Kudekar, "Design of low-density parity check codes for 5G New Radio," *IEEE Communications Magazine*, vol. 56, no. 3, pp. 28–34, March 2018.

[57] D. J. Costello and G. D. Forney, "Channel coding: The road to channel capacity," *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1150–1177, 2007.

[58] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge University Pr., 2005.

[59] S. Lin and D. J. Costello, *Error Control Coding (2nd Edition)*. Pearson, 2004.

[60] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, January 1962.

[61] R. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, Sep. 1981.

[62] A. Shokrollahi, "LDPC codes: An introduction," in *Coding, Cryptography and Combinatorics*, K. Feng, H. Niederreiter, and C. Xing, Eds. Basel: Birkhäuser Basel, 2004, pp. 85–110.

[63] L. H. C. Lee, *Convolutional Coding: Fundamentals and Applications* (Artech House telecommunications library). Artech House, 1997.

[64] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Transactions on Communications*, vol. 36, no. 4, pp. 389–400, April 1988.

[65] P. Frenger, P. Orten, T. Ottosson, and A. Svensson, "Multi-rate convolutional codes," Communication Systems Group, Department of Signals and Systems, Chalmers University of Technology, Tech. Rep., 1998.

[66] B. Moision, "A truncation depth rule of thumb for convolutional codes," in *2008 Information Theory and Applications Workshop*, Jan 2008, pp. 555–557.

[67] The Mathworks, Inc. (2019). Digital Modulation. [Online]. Available: https://www.mathworks.com/help/comm/ug/digital-modulation.html

[68] A. J. Viterbi, "An intuitive justification and a simplified implementation of the map decoder for convolutional codes," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 260–264, Feb 1998.

[69] C. E. Shannon, “A mathematical theory of communication,” *The Bell System Technical Journal*, vol. 27, no. 4, pp. 623–656, Oct 1948.

[70] J. Ha, J. Kim, and S. W. McLaughlin, “Rate-compatible puncturing of low-density parity-check codes,” *IEEE Trans. Inf. Theor.*, vol. 50, no. 11, pp. 2824–2836, Sep. 2006.

[71] X. Zhao, L. Zhu, Y. Guo, and X. Gou, “An effective puncturing algorithm for QC-LDPC codes with dual-diagonal structure,” in *Proceedings of 2012 5th Global Symposium on Millimeter-Waves*, May 2012, pp. 38–42.

[72] S. Choi, Y. Shin, J. Heo, K. Cho, and M. Oh, “Effective puncturing schemes for block-type low-density parity-check codes,” in *2007 IEEE 65th Vehicular Technology Conference - VTC2007-Spring*, April 2007, pp. 1841–1845.

[73] Y. Xu, Y. Wei, and W. Chen, “On the performance evaluation of quasi-cyclic LDPC codes with arbitrary puncturing,” in *2010 IEEE 71st Vehicular Technology Conference*, May 2010, pp. 1–5.

[74] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[75] A. Maltsev *et al.*, “IEEE 802.11-09/0334r8: Channel Models for 60 GHz WLAN Systems,” TGad Working Group, Tech. Rep., May 2010.

[76] A. Maltsev *et al.*, “IEEE 802.11-15/1150r9: Channel Models for IEEE 802.11ay,” TGay Working Group, Tech. Rep., March 2017.

[77] The Mathworks, Inc. (2019). wlanTGayChannel. [Online]. Available: https://www.mathworks.com/help/wlan/ref/wlantgaychannel-system-object.html

[78] J. P. Odenwalder, “Optimum decoding of convolutional codes,” Ph.D. dissertation, Dept. Syst. Sci., Sch. Eng. Appl. Sci., Univ. California, 1970.

[79] X. Chen, “Coding in 802.11 WLANs,” Ph.D. dissertation, Hamilton Institute, National University of Ireland Maynooth, 2012.

[80] D. Haccoun and G. Begin, “High-rate punctured convolutional codes for Viterbi and sequential decoding,” *IEEE Transactions on Communications*, vol. 37, no. 11, pp. 1113–1125, Nov 1989.

[81] M. Gast, *802.11ac: A Survival Guide*. O’Reilly Media, Inc., 2013.

[82] G. Breit, H. Sampath, S. Vermani *et al.*, “IEEE 802.11-09/0308r12: TGac Channel Model Addendum. Version 12.” TGac Working Group, Tech. Rep., March 2010.

[83] V. Erceg, L. Schaumacher, P. Kyritsi *et al.*, "IEEE 802.11-03/940r4: TGn Channel Models. Version 4." TGn Working Group, Tech. Rep., May 2004.

[84] The Mathworks, Inc. (2019). wlanTGacChannel. [Online]. Available: https://www.mathworks.com/help/wlan/ref/wlantgacchannel-system-object.html

[85] M. Luby, "Broadcast delivery of multimedia content to mobile users," Qualcomm Incorporated, Tech. Rep., 2013. Available: https://www.qualcomm.com/documents/broadcast-delivery-multimedia-content-mobile-users

[86] "RaptorQ technical overview," Qualcomm Incorporated, Tech. Rep., 2010. Available: https://www.qualcomm.com/documents/raptorq-technical-overview

[87] The Mathworks, Inc. (2019). 802.11ad packet error rate single carrier PHY simulation with TGay channel. [Online]. Available: https://www.mathworks.com/help/wlan/examples/802-11ad-packet-error-rate-single-carrier-phy-simulation-with-tgay-channel.html

[88] The Mathworks, Inc. (2019). Estimate BER for hard and soft decision Viterbi decoding. [Online]. Available: https://www.mathworks.com/help/comm/ug/estimate-ber-for-hard-and-soft-decision-viterbi-decoding.html

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California