TECHNISCHE
UNIVERSITÄT
DARMSTADT

COLLIDE, COLLATE, COLLECT: RECOGNIZING SENDERS IN WIRELESS COLLISIONS

JOHANNES TOBIAS LAUINGER

Bachelor Thesis

August 10, 2017

Secure Mobile Networking Lab
Department of Computer Science

Collide, Collate, Collect: Recognizing Senders in Wireless Collisions
Bachelor Thesis
SEEMOO-BSC-0103

Submitted by Johannes Tobias Lauinger
Date of submission: August 10, 2017

Advisor: Prof. Dr.-Ing. Matthias Hollick
Supervisor: Robin Klose, M.Sc.

Technische Universität Darmstadt
Department of Computer Science
Secure Mobile Networking Lab

ABSTRACT

With wireless mobile IEEE 802.11a/g networks, collisions are currently inevitable despite effective counter measures. This work proposes an approach to detect the MAC addresses of transmitting stations in case of a collision, and measures its practical feasibility. Recognizing senders using cross-correlation in the time domain worked surprisingly well in simulations using Additive White Gaussian Noise (AWGN) and standard Matlab channel models.

Real-world experiments using software-defined radios also showed promising results in spite of decreased accuracy due to channel effects. During the experiments, various Modulation and Coding Schemes (MCSs) and scrambler initialization values were compared. Knowledge about which senders were transmitting leading up to a collision could help develop new improvements to the 802.11 MAC coordination function, or serve as a feature for learning-based algorithms.

ZUSAMMENFASSUNG

In drahtlosen mobilen Netzwerken nach den IEEE 802.11a/g Standards sind Kollisionen trotz wirkungsvoller Gegenmaßnahmen nicht vollständig zu vermeiden. Diese Arbeit stellt einen Ansatz zur Erkennung der MAC-Adressen der beteiligten Sender bei einer Kollision vor und untersucht, inwiefern das Verfahren in der Praxis funktioniert. Über Kreuzkorrelation im Zeitbereich funktionierte die Erkennung in Simulationen unter Additivem Weißen Gaußschen Rauschen (AWGN) und verschiedenen Standard-Kanalmodellen von Matlab erstaunlich gut.

Praktische Experimente mit Software-Defined Radios zeigten ebenfalls vielversprechende Ergebnisse, wenn auch die Genauigkeit der Erkennung durch Kanaleffekte beeinträchtigt wurde. Bei den Experimenten wurden verschiedene Modulation and Coding Schemes (MCSs) und Scrambler-Initialisierungen verglichen. Die Kenntnis über die beteiligten Sender bei einer Kollision könnte zur Verbesserung der Koordinierungsfunktion oder als Feature für lernbasierte Verfahren verwendet werden.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACK	Acknowledgment
ACR	Active Collision Recovery
ADC	Analog-to-Digital Converter
AFD	Asynchronous Full Duplex
AGC	Automatic Gain Control
AP	Access Point
ARP	Address Resolution Protocol
AWGN	Additive White Gaussian Noise
BCC	Binary Convolution Code
BPSK	Binary Phase Shift Keying
BSSID	Basic Service Set Identifier
CRC	Cyclic Redundancy Check
CSMA	Carrier Sense Multiple Access
CSMA/CA	CSMA with Collision Avoidance
CSMA/CD	CSMA with Collision Detection
CSMA/CR	CSMA with Collision Resolution
CTS	Clear To Send
DCF	Distributed Coordination Function
DIFS	DCF Inter-Frame Space
FCS	Frame Check Sequence
FEC	Forward Error Correction
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
IEEE	Institute of Electrical and Electronics Engineers
IFFT	Inverse Fast Fourier Transform
LFSR	Linear Feedback Shift Register
LQI	Link Quality Indicator
LTF	Long Training Field

MAC	Media Access Control
Mbps	Megabits per Second
MCS	Modulation and Coding Scheme
MIMO	Multiple Input, Multiple Output
NAV	Network Allocation Vector
OFDM	Orthogonal Frequency Division Multiplexing
PHY	Physical Layer
PLFC	Packet Level Failure Classification
PSDU	Physical Layer Service Data Unit
QAM	Quadrature Amplitude Modulation
RMS	Root Mean Square
RSSI	Received Signal Strength Indicator
RTS	Request To Send
SDK	Software Development Kit
SDR	Software-Defined Radio
SEEMOO	Secure Mobile Networking Lab
SIC	Successive Interference Cancellation
SIFS	Short Inter-Frame Space
SIG	Signal Field
SISO	Single Input, Single Output
SNR	Signal-to-Noise Ratio
SRV	Service Field
STF	Short Training Field
USRP	Universal Software Radio Peripheral
WARP	Wireless Open-Access Research Platform

1 INTRODUCTION

1.1 MOTIVATION

In today's modern society, fast mobile networking has become indispensable. A huge and ever growing number of devices depend on, or at least support, wireless local area networks. Most of them follow a protocol from the IEEE 802.11 standards family. As technology improves, new standards are added and enhancements to the existing ones are proposed, making the network faster and more reliable. We have become accustomed to using such networks for a variety of activities, including mobile video streaming or wireless Internet access on laptops.
There are a number of known problems with the IEEE 802.11 protocols. One of them is the possibility of data frames colliding due to the random access scheme [4]. A collision occurs when two stations transmit at the same time, resulting in an illegible frame at the receiver. Although previous research has suggested some techniques to reduce or even decode collisions, as described in the related work section, the receiver often has to disregard the frame and wait for a retransmission.

While it is quite easy to merely detect the existence of a collision most of the time, detecting which exact stations were transmitting is considered difficult [6, 14]. However, this information could be beneficial for the operation of the network in multiple ways.
First, new coordination functions could be developed, which leverage the knowledge about colliding senders to determine which station should transmit next. This could for example mean that instead of retransmitting a frame when the acknowledgment from the receiver is missing, a station would pause sending for some time. All other stations on the network could compute the next sender, potentially preventing a new collision from occurring if the condition that led to the initial problem is somehow correlated to that sender.
Second, it could be used to create statistics about which senders participate in collisions more often than others, potentially relating this information with parameters like hardware vendor, operating system, physical location, or others. This could also be done in real-time, providing a means to monitoring a network and thus allowing the administrator to automatically exclude stations that cause disturbances in the network.

In this thesis, I propose a technique to recognize the senders of a collided frame based on sample cross-correlation in the time domain, and evaluate it using both simulations and software-defined radios.

1.2 CONTRIBUTIONS

The main contributions of this thesis are:

1. *Design and Implementation of a MAC Address Recognition Technique*

   I describe a design for an algorithm that allows to detect transmitting stations involved in a collision at the receiver. The technique uses a cache of MAC addresses as seen in the network to pre-compute time-domain representations of frames. When a collision occurs, complex samples are cross-correlated to the available signal pool to detect the most likely collided nodes.

   The algorithm is implemented as a proof-of-concept in Matlab. All simulations are based on that code.

2. *Evaluation through Simulation and WARP SDRs*

   I measure the detection accuracy and performance in simulations that vary different parameters of the transmitted frames related to the MAC header field. Furthermore, several channel models are applied to evaluate the resilience to interference and attenuation.

   Finally, I use Wireless Open-Access Research Platform (WARP) Software-Defined Radios (SDRs) to find out how well the technique works in a real-world scenario.

1.3 OUTLINE

This thesis is structured as follows: Chapter 2 provides an introduction to important parts of the IEEE 802.11 standard, as well as background information on the relevant mathematical and physical concepts. I give an overview on and comparison to related work in Chapter 3. Chapter 4 covers the design and implementation of my proposed algorithm to recognize sender MAC addresses. This technique is evaluated, and results are presented in Chapter 5. I discuss implications of the results, problems, and possible future work in Chapter 6. Finally, I conclude this work in Chapter 7.

2 BACKGROUND

This chapter provides an introduction to the overall structure of the IEEE 802.11 standards for wireless networks. It covers the essential parts to recognizing sender MAC addresses. Furthermore, background information on cross-correlation is given.

2.1 IEEE 802.11 MAC AND PHY

The IEEE 802.11 standards family describes wireless local area networks as supported by all current major operating platforms. They inherit the reference structure of their parent standard, IEEE 802. Specifically, they share the MAC address. This is a six byte device address used to identify a physical network interface.
The standard is divided into the Physical (PHY) and the Medium Access Control (MAC) layer specifications.

2.1.1 *Physical Layer Specification*

There are a number of different physical layer specifications available for use in IEEE 802.11 networks, which define various parameters such as the frequency used for transmissions.
In this work, I only use the IEEE 802.11a/g standards. These use Orthogonal Frequency Division Multiplexing (OFDM) to distribute bits over a channel spectrum. The maximum achievable data rate is 54 Megabits per Second (Mbps) [1]. The most significant difference between the two is that 802.11a uses a carrier frequency of 5 GHz, whereas 802.11g transmits at 2.4 GHz. Around the carrier frequency, multiple channels of 20 MHz bandwidth are available. These Physical Layer (PHY) standards are extremely widely supported but were superseded by the 802.11n specification.
With 802.11n, being the first high-throughput standard, channels with more bandwidth and multiple spatial streams, also known as Multiple Input, Multiple Output (MIMO), were introduced, providing a much higher data rate. 802.11ac and the currently in-development 802.11ax improve even further on that. To evaluate sender detection techniques in practice, the 802.11g PHY is considered in this work, which is fundamental for later versions of the standard. Future work could adapt the technique to 802.11n/ac/ax.

The PHY frame structure comprises several fields. They are illustrated in Figure 1. The Physical Layer Service Data Unit (PSDU) is the payload which is passed down for transmission by the MAC layer. It is encapsulated within the 16 bits Service Field (SRV) in the beginning, and six tail bits and additional padding bits in the end. The combination of these is called Data Field.
The Data Field is modulated using the following process: First, the bits are scrambled by applying XOR with a synchronous bit sequence. This scrambling sequence is repeatedly generated by a Linear Feedback Shift Register (LFSR) derived by the polynomial $G(D) =$

$D^7 + D^4 + 1$, using a seven bit state register [1]. When transmitting, the initial scrambler state should be set to a pseudo-random non-zero state for each frame [1]. Since the Service Field, as described earlier, is prepended with all bits set to zero, the first seven bits contain the initialization value after scrambling. This is because for any value A, it holds $A \oplus 0 = A$. When receiving a frame in normal operation, the service field is used to synchronize the descrambler to that same initialization.

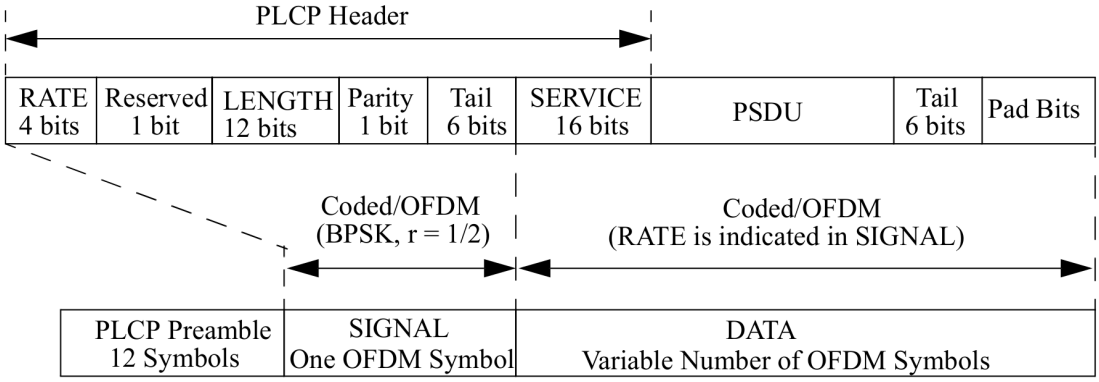


Figure 1: IEEE 802.11 Physical Layer Frame Structure [1]

After that, convolutional encoding is applied to the bits. The standard allows the encoder to use different code rates as specified by the Modulation and Coding Scheme (MCS). The encoder also uses a seven bit state [21]. Bits are now grouped into symbols and rearranged using a deterministic permutation function [23]. This is called interleaving. Following this, the symbols are modulated using Quadrature Amplitude Modulation (QAM) with different possible bit rates.

The available constellations are described by the MCS. Table 1 shows all available MCSs for IEEE 802.11a/g. Finally, pilot signals are added, the signal is transformed into the time domain using the Inverse Fast Fourier Transform (IFFT), a cyclic prefix is added, and the signal is further processed and sent.

MCS	**Modulation**	**Coding Rate**	**Coded bits per symbol**	**Data bits per symbol**
0	BPSK	1/2	48	24
1	BPSK	3/4	48	36
2	QPSK	1/2	96	48
3	QPSK	3/4	96	72
4	16QAM	1/2	192	96
5	16QAM	3/4	192	144
6	64QAM	2/3	288	192
7	64QAM	3/4	288	216

Table 1: IEEE 802.11a/g Modulation and Coding Schemes (MCS) [1]

The Signal Field (SIG), which is transmitted before the Data Field, contains the MCS used for the frame, and its length in octets. The Signal Field is always modulated with binary phase shift keying (BPSK) and rate $\frac{1}{2}$ Binary Convolution Code (BCC). Nearby stations will also decode the SIG to defer their transmissions for an appropriate time. The Signal Field has a duration of 4 µs.

In order to allow the receiver to calculate path effects (see section 2.4) of the channel, and to equalize received data, training fields are used in the beginning of the frame. The Short Training Field (STF) is used for start-of-frame detection and Automatic Gain Control (AGC). It includes 10 repetitions of a 0.8 µs symbol, resulting in a duration of 8 µs. The sequence is chosen to have good correlation properties and a low peak-to-average power, meaning its properties are preserved after clipping [23].
The Long Training Field (LTF) is used for channel estimation, more precise frequency offset estimation, and time synchronization. It comprises two repetitions of a 3.2 µs training symbol, and 1.6 µs cyclic prefix, therefore the LTF also spans 8 µs. Both training fields and the Signal Field form the preamble. Figure 2 shows detailed timing information for this.

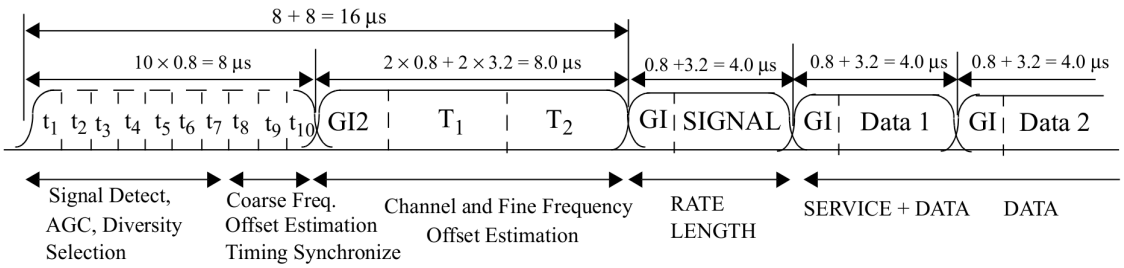


Figure 2: IEEE 802.11 Preamble Timing [1]

2.1.2 *Medium Access Control Layer Specification*

The Medium Access Control Layer (MAC) defines which station may transmit at a given time. The IEEE 802.11 MAC layer uses CSMA with Collision Avoidance (CSMA/CA). With this algorithm, every station listens to the medium before transmitting. Collisions are prevented if possible, and detected through the lack of an acknowledgment frame. In contrast, Ethernet uses CSMA with Collision Detection (CSMA/CD), where a transmitting station can directly sense a collision [2]. This is not possible with wireless communication because stations generally can not transmit and receive at the same time.

A MAC frame includes a MAC header, a variable-length body, and a Frame Check Sequence (FCS) containing a Cyclic Redundancy Check (CRC) code.
The MAC header consists of two bytes frame control, two bytes frame duration in microseconds, 18 bytes address 1, 2 and 3, as well as some others depending on the frame control field. In a data frame, the address fields contain the destination address (DA), source address (SA), and the Basic Service Set Identifier (BSSID). The frame structure is illustrated in Figure 3.

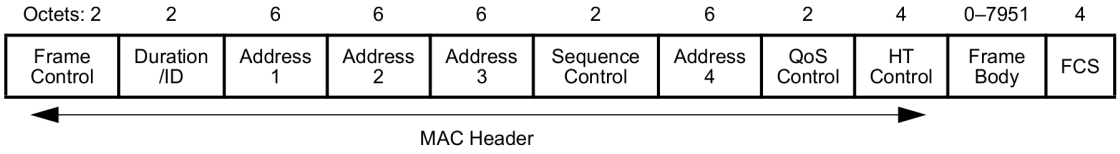


Figure 3: IEEE 802.11 MAC Layer Frame Structure [1]

2.2 DISTRIBUTED COORDINATION FUNCTION

The Distributed Coordination Function (DCF) precisely describes when sending stations may transmit. It logically belongs to the MAC layer. A station may send after sensing the medium idle, or at least one DCF Inter-Frame Space (DIFS) after the medium has become idle [4]. While a sender has access to the medium, it separates frames by a Short Inter-Frame Space (SIFS), and other stations will not capture the medium because they must wait for a longer DIFS.
After sensing that the medium is idle, a station must wait a random backoff time before transmitting. This helps to reduce collisions caused by multiple stations waiting to transmit. The backoff time is pseudo-randomly chosen out of the interval $[0..CW]$, where CW is the so-called contention window. On each unsuccessful transmit, CW is doubled until it reaches a maximum value. It is reset to its initial value as specified by the standard after a successful transmission.

The IEEE 802.11 MAC uses positive frame acknowledgments by layer 2 Acknowledgment (ACK) frames. Without an ACK the sender retransmits the frame. To further reduce collision impact, large payloads are fragmented into multiple PSDUs, which are acknowledged individually. Data frames include a retry bit in the sequence number field to detect duplicate frames [23].

The Hidden Node problem can occur when a station sees one station, for example an Access Point (AP), but not a third station that is currently transmitting to the AP. The medium is busy, but the first station senses it as idle and begins a transmission. Therefore, the transmission to the AP is subject to a collision. One approach to mitigate this is to use an additional handshake mechanism before transmitting.
Before sending the data frame, a station transmits a Request To Send (RTS) Frame containing the desired sending duration. The access point will reply with a Clear To Send (CTS) Frame, and third stations update a Network Allocation Vector (NAV) to remember the busy medium. In the hidden node scenario, the offending station would be able to receive the CTS frame from the AP, and is therefore aware of the ongoing transmission. On top of that, RTS frames are much shorter than data frames, meaning a collision is less harmful to the medium because less air-time is wasted.

2.3 CROSS-CORRELATION

The cross-correlation is a function of two continuous or discrete value series that measures their similarity [24]. It is often used to compare different functions, or to search for a small feature in a larger stream of data.

With IEEE 802.11, cross-correlation is applied to find the beginning of frames by correlating a known symbol of the Short or Long Training Field with a sample stream as captured by a receiver. However, in this thesis I will also use it to correlate the time-domain representation of MAC addresses.

Since wireless transmission samples are discrete values, I disregard the continuous form of cross-correlation here. The discrete variant for series of samples f and g is calculated as follows:

$$(f \star g)(n) = \sum_{m=-\infty}^{\infty} f^*(m)g(m+n)$$

This formula is related to the convolution of the functions, but it features the complex conjugate f^* of the first function for the sum of products. A graphic analogy for cross-correlation would be sliding the functions against each other, where for each point all function values are multiplied individually and then added up.
In general, higher values of cross-correlations mean that the functions are more similar. It is worth noting that periodic functions have periodic cross-correlations, and that the IEEE 802.11 Long Training Field is designed specifically in such a way that correlation is many orders of magnitude higher if the samples of two copies overlap exactly, than it is for any other alignment [23].

2.4 MULTI-PATH CHANNEL EFFECTS

In an ideal scenario, a transmitted signal would travel on a direct line-of-sight from sender to receiver. However, electromagnetic waves spread in a spherical shape. In a real-world situation, signals can therefore reach the receiver on multiple paths, each affected by different delay, attenuation, reflection, or other effects.
A common multi-path effect are echoes, which are caused by signals being reflected on walls, furniture, or similar obstacles. When an echo occurs, an attenuated copy of the signal with a delay is received.

When trying to recognize senders in collisions, it is important to take multi-path interference into account. Especially problematic is the fact that the signals add up at the receiving antenna. This means that the training fields in the preamble are affected by different paths and can hardly be used to calculate the inverted channel matrix. Chapter 5 provides more details on this problem.

3

RELATED WORK

In this chapter I present related research and relevant previous work. The later proposed technique of recognizing senders in a collision by correlating their MAC addresses in the time domain has to my best knowledge not been described in the literature yet. However, there have been experiments which point towards that idea. This chapter is organized in sections dealing with the coordination function, hidden terminals, collision handling, and consequences of the capture effect.

3.1 IMPROVEMENTS TO AND ALTERNATIVE DCFS

As described in section 2.2, the Distributed Coordination Function (DCF) defines the rules of communication in an IEEE 802.11 network. There are measures to reduce the impact of a collision, such as the two-way handshake using Request To Send (RTS) and Clear To Send (CTS) frames. This procedure allows stations in the network to request transmission for a specific time. Other stations can listen to this RTS and update their Network Allocation Vector (NAV) tables accordingly. If a collision occurs, it will happen on the RTS frame. Since the RTS frame is much shorter than a data frame, the amount of unusable air-time is smaller, as less data has to be discarded.
Giuseppe Bianchi contributed a performance analysis of the DCF [4]. The maximum load that the system can carry in stable condition is called saturation throughput. The author develops a Markov model to derive throughput estimations and verifies them against simulations. A result is that using the two-way handshake is not effective in practice. The throughput did not increase in experiments which are comparable to a typical wireless application, mostly due to the overhead introduced by the additional RTS and CTS frames. Therefore, RTS and CTS frames are not used in most network configurations nowadays [4, 6, 8].

ReCoder [18] is an algorithm for neighbor discovery in 802.11 networks that does not use beacon frames. Instead, a new frame format that only needs to be correlated, not decoded, is introduced. The technique saves energy and is more resilient to collisions, since the otherwise necessary decoding of the frame is not possible in that situation.
These alternative frames start with a fixed, independent training sequence called RCover. This sequence is used to identify a neighbor discovery message, similar to frame detection using the Short Training Field. After that, a two-stage identification pattern is applied. The first stage is a fixed Gold-code signal with low off-peak autocorrelation, shifted by a unique amount of samples. These are chosen randomly by the sending station. In the event that two stations choose the same offset, they are distinguished by the second stage, which is a hash of the MAC address. This stage is less distinguishable and therefore only acts as a tie-beaker for identical first stages.

Receivers are triggered by the RCover sequence and correlate the first identity stage to the list of known stations. If a match is found, the second stage is checked to eliminate possible duplicate shift-amounts. If however no match is found, the reference table is adjusted in a way that the new identity is now included.

As Zhu et al. pointed out, real-world packet loss can be caused by both a weak link, and collisions. These conditions can change very fast [31]. Reasons for this include humans being in the signal path, external interference, and unpredictable transmissions by other stations. Distinguishing failure conditions must thus be done at the packet level to provide useful information. Both naive assumptions of only collisions, or only weak links, cause a waste of resources, i.e. air-time or transmission power.
The authors present a new algorithm, Packet Level Failure Classification (PLFC), which collects a byte-level Received Signal Strength Indicator (RSSI) and a packet-level Link Quality Indicator (LQI). Using this information, the receiver provides the sender with an elaborated retransmission request. This includes whether there was a weak link or a collision. The sender can then for example increase transmission power to mitigate a weak link, or just retransmit if a collision occurred [31].

For continuous and fairly regular data streams from multiple clients, such as in an environment with several VoIP transmissions, the 802.11 MAC exponential backoff can lead to some clients experiencing lower throughput [16]. Lin et al. presented Lock Step, an algorithm introducing proactive backoff times after successful transmissions. This allows other clients to step in and transmit their data. With Lock Step, clients extend the carrier sense mechanism by measuring the percentage of free air-time within eight continuous slots, and adjusting their transmission behavior based on that fraction. After some collisions have occurred, all N clients will eventually be transmitting every 8N slots, thus providing a fairer distribution to all stations in the network [16].

While the ReCoder algorithm provides sender detection, it requires a change to the standard IEEE 802.11 MAC layer, adding new frame types. This is a shared trait with PLFC. Therefore, the technique must be implemented in all devices in order to benefit from it. This is a huge disadvantage since it requires high timing precision, which may not be available for custom modifications on off-the-shelf transceiver hardware. One approach to handle such modifications is a split-layer architecture as proposed by [20], which uses the host's CPU for some tasks and therefore provides flexibility while maintaining timing constraints. This thesis however tries to detect senders in a collision without any changes to the MAC layer and is therefore different.

Similar to a possible usage scenario of this thesis, previous work has suggested assigning a priority list for retransmissions after a collision. CSMA with Collision Resolution (CSMA/CR) [6] grants one station priority access to retransmit, so that there will not be a follow-up collision. [30] proposed an algorithm that allows the receiver to reschedule all collided frames: Each sender picks a random sequence of samples. If a collision is detected by the receiver, it broadcasts a request for the missing sequences. These sequences are transmitted in parallel, deliberately causing a collision. The receiver then decodes them using cross-correlation and broadcasts a scheduling frame to all identified

senders at once. The senders are assigned a slot in which they resend their frame in a well-defined queue, not losing any air-time by unused slots, as would occur with the binary exponential backoff algorithm of CSMA with Collision Avoidance (CSMA/CA). The parallelism is possible by choosing sample sequences with low self-correlation properties.

3.2 HIDDEN TERMINALS

Hidden Terminals occur in situations where CSMA/CA as employed by the IEEE 802.11 MAC layer can not correctly detect an ongoing transmission. This can happen when a receiver is in range of two stations, but these stations are not able to reach each other [23].
In this case, it is possible that one sender does not recognize a transmission of the other sender. It therefore starts sending and disturbs communications between the other sender and the receiver. This effectively renders both transmissions lost as the frames can not be decoded by the receiver anymore.

CSMA/CA is often over-protective, meaning there is some air-time that is not used for transmissions. This is due to the fact that carrier sensing is done by the sender. A better approach could be to look at current conditions at the receiver [11]. These determine whether or not the receiver will be able to decode the frame. If two stations are hidden from each other, even multiple transmissions could be conducted at the same time for the right sender and receiver pairs, providing improved spatial efficiency.

A different approach is to use full-duplex transmissions to continuously provide feedback whether the receiver is able to decode the frame. This technique is proposed by Vermeulen et al. [26]. While receiving a frame, a station keeps transmitting an acknowledgment. If at some point a collision occurs, the receiver stops sending this acknowledgment. The sender then detects this and cancels the transmission. The simultaneously transmitted acknowledgment frame thus acts like a continuous CTS frame, letting other stations know of the transmission and partially solving the hidden node problem. Other research has concluded that coordination time can be eliminated using to some extent full-duplex communication. The authors call their approach Asynchronous Full Duplex (AFD) [17]. With AFD, carrier sensing and contention for upcoming air-time is done while data is being transmitted.
Cao et al. presented different scenarios regarding exposed terminals, and proposed an algorithm to maximize throughput while minimizing collisions in the case of some exposed terminal situations. The authors apply behavioral learning based on incoming traffic patterns [5]. While exposed terminals are a rather specific scenario, behavioral learning could be an interesting topic for further research. Based on the detected frequency of a specific sender in collisions, improvements to the coordination function may be possible.

3.3 COLLISION DETECTION AND DECODING

When CSMA/CA fails, such as with hidden terminals, it is possible that senders repeatedly collide, or one completely captures the medium. The RTS / CTS handshake mechanism is commonly not used as a counter measure due to its negative effects on throughput [4, 6, 8]. Gollakota et al. observed that senders tend to collide again on the exact same frames, and due to jitter collisions start with a random interval of delay. The authors proposed a new technique called ZigZag. With ZigZag, a station finds a part of the collided frame that is collision-free in one instance, and subtracts that from the collision in a second instance. This allows the receiver to decode a small part of the frame. This chunk can then again be subtracted from the first collision, yielding more decoded parts. After a number of repetitions, the entire frame is decoded.
Using ZigZag, senders do not need to make a trade-off between resilience to collisions and throughput, instead they transmit at the network's saturation rate. When no collisions occur, the frame demodulation process remains unchanged. However, in the presence of a collision ZigZag can manage to decode it and therefore maintains a high information rate in the network. ZigZag is independent from the used Modulation and Coding Scheme (MCS), backwards-compatible to the standard IEEE 802.11 MAC layer, and even generalizable to a collision with more than two frames.

Detecting the start of frames in the collision is done by correlating the signal with the known 802.11 preamble. This correlation is done on a shifting start sample, such that the highest correlation value will indicate the sample at which the preamble starts. Correlation is almost zero except when the preamble is perfectly aligned [1]. More than one spike in the correlation shows the presence of a collision, and also allows the receiver to detect the offset between the collided frames. I will use the same technique to detect collision within the proof-of-concept implementation for this thesis.

Stripping decoder (Strider) [9] is a rate-less code that can be used in the IEEE 802.11 MAC. Strider strives to be collision-resilient, meaning that senders do not need to care for collision as receivers will be able to decode frames even after suffering from a collision. For this, a minimum distance transformation (MDT) is used [9]. Modulated symbols are mapped to a larger symbol space, such that the minimum distance between symbols is larger than normally required. Hence, Strider automatically achieves the best code rate for a given channel.
AutoMAC [10] is a modified MAC layer. It uses the Strider code, although other rateless codes would be possible, and embraces collisions. As avoiding them is no longer necessary, simpler MAC implementations are possible. A similar proposed change to the MAC layer is Mozart [3], which also encourages collisions and queues retransmissions in an intelligent way so that the receiver will then be able to decode all frames.
This work is interesting, but not directly applicable to this thesis. Strider would probably allow to decode colliding MAC addresses, but only if the senders are a-priori using the code. This thesis tries to achieve its goal without requiring any changes to the standard MAC layer.

A different scenario where collisions can happen is when broadcast frames get relayed by intermediate stations [12]. Decoding such broadcast collisions would provide much better throughput. However, previously described collision recovery techniques can not be used here. ZigZag [8], for example, requires that frames must be retransmitted at least once. This is not the case with non-acknowledged, only relayed, broadcast transmissions. Successive Interference Cancellation (SIC) [22] demands that one frame is known to the receiver, which is also not possible here.
Onion Decoding [27] and Chorus [29] use an iterative approach comparable to ZigZag, where collisions are detected by correlating the preambles. The Onion Decoder needs an offset between the colliding frames, and the first (collision-free) chunk is subtracted from the collision to obtain a new collision-free chunk. However, Onion Decoding does not require a second collision of the same frame, because it is specifically designed for broadcast collisions, where multiple stations are forwarding the broadcast frame. Hence, both frames in the collision are the same.
Active Collision Recovery (ACR) [28] tries to make collisions occur in a so-called Long Short (LS) form, meaning a long frame is colliding with a short frame. This comes with some advantages. For example, receivers can still decode many chunks from the long frame, and ACR can then employ Forward Error Correction (FEC) to decode the entire frame. Collisions are detected by calculating a Cyclic Redundancy Check (CRC) code. When the checksum is wrong, a collision is assumed.
On the MAC layer, payload data from the network layer is actively shaped in long and short frames which are passed to the Physical Layer (PHY) for transmission. The MAC layer decides randomly whether to first send the short or the long frame. Furthermore, the Carrier Sense Multiple Access (CSMA) algorithm is modified to increase the likelihood of collisions to be in LS form. This is achieved by a non-uniform distribution of contention slots and a transmission delay for short frames [28].

Keene et al. [14] presented an algorithm to locate the part of a frame that is not decodable due to a collision. The algorithm uses a likelihood ratio test to determine whether a symbol suffered from an error. It then determines the frame chunk with the largest number of errors using a sliding window approach. In some cases, the frame can be fully decoded by correcting the bit errors using an error resolving code.

3.4 CAPTURE EFFECT

The capture appears when due to differences in signal power, a receiver physically focuses on one specific transmission. Other incoming signals are treated as noise [15]. In the case of a collision, this means that the signals do not simply add up together, but it is possible that one transmission is received, while the other one is barely noticed.

Park et al. suggested a mechanism that detects a frame with high power interrupting an ongoing reception of a frame with lower power. This is then counted as a collision. The amount of these collisions is shared using a new frame type, informing neighboring stations about the situation. These stations then calculate the sum of nearby collisions and adjust the code rate to use [21].

Similar to [31], [7] tries to decide whether a frame decoding error was due to a collision or weak link by examining the power level at the receiver. The authors observe that when a collision occurs, the power level is not simply the sum of the two transmissions, but rather includes some form of jitter due to oscillations in Automatic Gain Control (AGC) and other hardware with feedback loops.

Jibukumar et al. proposed a protocol called Receiver-Initiated Fast Sequential Collision Resolution (RFSR) [13]. It is somewhat similar to CSMA/CR [6] in that when a station detects a collision, it sends out a jamming signal to inform all transmitters that they need to stop transmitting. After a collision, the collided frame is assigned priority access for retransmission.

As described in section 3.1, these techniques have the disadvantage of requiring a change to the standard IEEE 802.11 MAC layer. In contrast, this thesis tries to recognize colliding senders exclusively by changes to receiver implementations, not the coordination function itself.

4 DESIGN AND IMPLEMENTATION

This chapter proposes a method of detecting MAC addresses through sample correlation. The algorithm is implemented in Matlab as a proof-of-concept.

4.1 HIGH-LEVEL SENDER DETECTOR DESIGN

The high-level design of a sender detector based on MAC addresses is illustrated in Figure 4. From the beginning of my work until the final version, multiple design choices changed. These and their reasons are depicted in detail in Chapter 5. The overall concept has however remained unmodified.

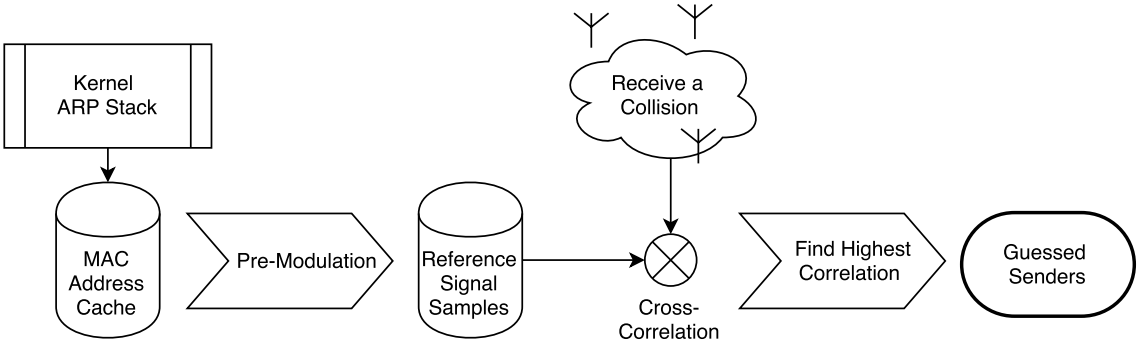


Figure 4: High-Level Detector Design Schema

First of all, it is important to decide which MAC addresses should be tested when processing a received collision. Since MAC addresses are six byte values, there are $2^{48} \approx 3 \cdot 10^{14}$ possible instances. Although some MAC addresses are invalid because the first three bytes denote the network interface vendor, and not all values have been issued[1], there are still by far too many to try out all of them for every collision.
Instead, I make use of the operating system's Address Resolution Protocol (ARP) stack. During normal network operation, the kernel already keeps a cache of MAC addresses that have been observed. This list is perfectly suited for detecting senders, as it is quite likely that a collision occurs between senders that are already in the network for some time.

Next, I modulate IEEE 802.11 data frames for every cached MAC address. These are used as reference signals for correlation with incoming collisions. Only a small chunk of the reference frames contains the MAC address bits and is thus important for sender detection. I describe this region in detail in section 4.3.
As mentioned in section 2.1, there are different possibilities for frame encoding, namely the Modulation and Coding Scheme (MCS), the scrambler initialization, and the error-correcting convolutional encoding. Since the convolutional encoder uses a seven bit

1 http://standards-oui.ieee.org/oui.txt

state, only the MAC header field directly preceding the sender MAC address is relevant here. That field is the destination MAC address. In theory, all possible combinations could be modulated for the reference signals cache, however this would mean a total amount of more than 130 thousand candidates for every MAC address in the ARP pool. I evaluate the impact of these factors in chapter 5.
The amount of 130 thousand candidates, as mentioned above, is the product of three values: N_{MCS} denotes the possible Modulation and Coding Schemes, which are eight. $N_{Scrambler}$ describes the available scrambler initialization values. For a seven bit state where the value zero is invalid [1], these are $2^7 - 1 = 127$. N_{Dest} is the amount of possible states of the convolutional encoder, which are $2^7 = 128$.

$$N_{MCS} \cdot N_{Scrambler} \cdot N_{Dest} = 8 \cdot 127 \cdot 128 = 130048$$

The modulation process comprises the following steps:

1. Generate a MAC header with appropriate sender and destination MAC addresses
2. Apply the scrambler with specific initialization value
3. Run the convolutional encoder, which is deterministic
4. Group bits and interleave symbols
5. For a time-domain signal, apply Inverse Fast Fourier Transform (IFFT) and add a cyclic prefix

For correlation in the frequency domain, I omit step five. The Fast Fourier Transform (FFT) has to be applied to the received collision signal in this case. I have done experiments with frequency-domain detection and show the results of these in section 5.1.

When the receiver senses a transmission, I determine whether it is a collision using the IEEE 802.11 Long Training Field (LTF). Section 4.2 gives details about this process. This also measures a possible delay between the two frames.
Upon noticing a collision, all reference samples are correlated with the signal under test. The correlations are then sorted in descending order by correlation magnitude. The highest correlations make up for the best guess which senders are subject to the collision.

It is worth noting that the output of this algorithm is always a probabilistic result. The technique will never detect senders with full confidence, but rather return a distribution of likelihood over the cache of reasonable MAC addresses.

4.2 DETECTING FRAMES AND COLLISIONS

A receiver captures a continuous stream of samples, generated by its Analog-to-Digital Converter (ADC). It is necessary to employ some mechanism to detect the start of a transmission. In regular operation, IEEE 802.11 receivers use the known preamble for this by correlating the LTF to the incoming data in a sliding window approach [23].

This can be extended to recognizing collisions. Since the LTF is designed in a way that it has low off-peak autocorrelation [1], there will be twice as many correlation peaks in a collision of two frames. The distance between the two LTF pattern can even be used to calculate a possible delay between the two transmissions. An example of such correlation peaks in case of a collision is shown in Figure 5. Each frame causes two large spikes, and one spike with smaller magnitude. This is caused by the composition of the LTF with two repetitions of a symbol, and a cyclic prefix of half the symbol size.

Knowing when the frames start is important for sender detection. Since the MAC address is represented by only a small number of time-domain samples (see section 4.3), it is vital to correlate the correct chunk of the signal.

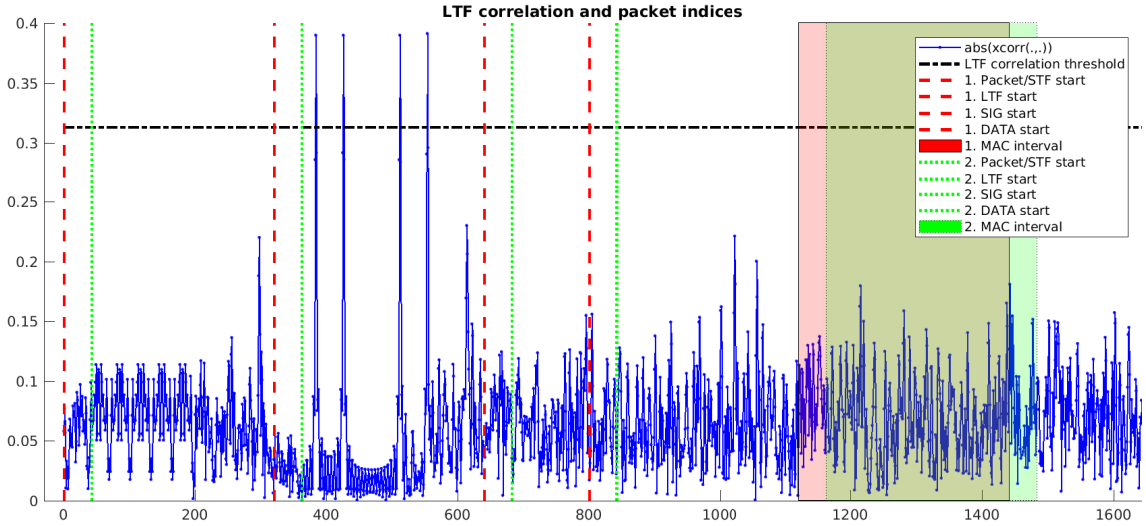


Figure 5: Cross-Correlation of Collision Samples with Long Training Field

Figure 5 offers additional markers for the different signal fields as described in section 2.1. Here, the symbols containing the MAC address of the first frame begin after 1120 samples. The second MAC address follows with an offset of about 40 samples.
When trying to decode these MAC addresses, the correlations with the reference pool signals obviously have to be applied to the combined region of the first and second MAC addresses. In this example, that would be all of the colored samples.

4.3 PERIODS CONTAINING MAC ADDRESSES

Depending on the MCS, the samples affected by different MAC addresses start at different positions. The byte offset within the MAC layer header remains the same. However, the amount of Orthogonal Frequency Division Multiplexing (OFDM) symbols that need to be skipped before the beginning of the MAC address, and the symbol count that has to be included to cover everything, change depending on the bit density.
Table 2 provides a summary of the various samples offsets for different MCSs. These values are calculated assuming a sampling rate of 20 MHz. Since IEEE 802.11a/g channels have a passband bandwidth of 20 MHz, this is the minimum required sampling rate [1]. However, some radios like the Wireless Open-Access Research Platform (WARP) boards

use a higher sampling rate of 40 MHz. In this case, all indices have to be doubled.

All frames begin with the preamble, specifically the Short and Long Training Fields, and the Signal Field. Both the Short Training Field (STF) and LTF have a duration of 8 µs. For the STF, this is 10 repetitions of a 0.8 µs symbol. For the LTF, there are two repetitions of a 3.2 µs symbol, and the 1.6 µs cyclic prefix. The Signal Field consists of one standard OFDM symbol with cyclic prefix. That is 4 µs in the case of IEEE 802.11.
In total, the preamble takes 20 µs. At 20 MHz sampling rate, this results in $20 \cdot 10^{-6}\text{s} \cdot 20 \cdot 10^{6}\text{Hz} = 400$ samples. The Data Field thus start at sample index 401. Within that, the Service Field as well as the Frame Control and Duration MAC header fields precede the MAC addresses.

MCS	**Address 1 (Destination)**	**Address 2 (Sender)**
0	561 - 720	721 - 880
1	481 - 640	561 - 720
2	481 - 560	561 - 640
3	401 - 560	481 - 560
4	401 - 480	481 - 560
5	401 - 480	401 - 480
6	401 - 480	401 - 480
7	401 - 480	401 - 480

Table 2: Sample Offsets of MAC Address Fields at Different MCSs

MCSs 0 and 2 have an important advantage. Only with these schemes does the above described region contain only the specific MAC address, since the encoded bits per OFDM symbol (24 for MCS 0, and 48 for MCS 1, as mentioned in Table 1) are integer multiples of the 48 bits MAC address length. All other schemes contain additional data before the address, behind it, or in both positions.

4.4 MATLAB IMPLEMENTATION

I built a proof of concept implementation of the presented technique in Matlab. Matlab was initially chosen for its large number of suitable toolboxes, and an existing implementation of the 802.11 MAC and PHY. It also allows to use WARP Software-Defined Radios (SDRs). The WARPLab 7.5.1 Matlab Software Development Kit (SDK)[2] provides easy-to-use high-level access to the radios. It was my preferred choice over the alternative of using Python and Universal Software Radio Peripheral (USRP) boards.

The Matlab code is structured into different sections. First, utility functions implement the generation of MAC headers, modulating IEEE 802.11 data frames in the time domain, and choosing sender MAC addresses randomly from a list. There are two different

2 http://warpproject.org/trac/wiki/WARPLab

libraries used for modulation: MathWorks' WLAN System Toolbox[3] as well as a custom IEEE 802.11 implementation developed at the Secure Mobile Networking Lab (SEEMOO) at TU Darmstadt.
A number of simulation scripts measure the detection algorithm's performance under simulated channel effects. The results of these are presented in chapter 5. A WARPLab script effectively does the same thing, but using three WARP SDRs. Finally, there are scripts that run a series of experiments varying certain parameters, save the data, and generate figures for evaluation.

The core of the Matlab implementation is cross-correlation. Using the Signal Processing Toolbox[4], this can be done using the `xcorr` function. This function expects two complex sample series and returns a complex correlation vector and a real vector of offsets. The offsets range from the negative feature length up to the positive length. This is because the data series can be slided against each other in both directions. Using this function is similar to this example for an LTF symbol:

```
1 [correlation, offset] = xcorr(rx, lts_t);
```

Listing 1: Cross-Correlation of an LTF Symbol

Analyzing the performance of my implementation revealed that most time was not spent on cross-correlation, but on the generation of the reference signal pool. I improved this code by applying the following optimizations.
First, since the sender detection only relies on correlating those samples that contain the MAC address, as described in section 4.3, most of the generated signal is discarded anyways. This means that it is unnecessary to spend valuable time on calculating the MAC layer checksum, which is stored after the payload data. I simply use a placeholder value of 0x42424242 for the checksum. Furthermore, I do not generate the preamble, namely the Training and Signal Fields. This can be done by using the `wlanNonHTData` function from the WLAN System Toolbox. This function merely modulates the Data Field, in contrast to the `wlanWaveformGenerator` function.
Second, I store the reference signals in a cache that is reused in as many consecutive experiments as possible. All experiments were repeated for at least 100 runs. Since the reference pool remained the same for all repetitions, using a shared variable for the entire set almost cut the cost of a higher sample size to zero. This is because in relation to modulating a signal, calculating cross-correlations consumes several orders of magnitude less time (about 0.2 ms compared to 650 ms). This is good news because in a real-world application, the generation of reference signals can be done ahead of time, and signals under test do not have to be modulated since they are received already in the time domain.
Incorporating these optimizations reduced the runtime of the experiment varying the scrambler initialization, which is presented in section 5.5, from an extrapolated 300 days to about 45 minutes on a consumer laptop. This not only allowed me to run the experiments at all, but also let me increase the number of repetitions, and therefore largely

3 https://www.mathworks.com/products/wlan-system.html
4 https://www.mathworks.com/products/signal.html

enhancing the sample size.

To simulate channel effects, different channel models from both the WLAN System Toolbox and the Communications System Toolbox [5] were used. The exact functions are described in the corresponding sections in chapter 5.

As addressed in section 4.2, it is necessary to detect the offset between two frames in a collision. This is done using the `meshgrid` Matlab function. Listing 2 shows an example of calculating the field indices of both collided frames. The code compares the signal-to-training symbol correlation with a defined threshold and finds peaks in the data. These are then grouped into four clusters. This effectively removes noise and measuring inaccuracies causing a spike to last for more than one sample. Due to the correlation of the preamble with one LTF symbol, there should be two large and one small spike. This is because as described in section 2.1, the LTF comprises two symbol repetitions and a cyclic prefix of half a symbol length. Since the correlation threshold is chosen to be between the small and large spike, and assuming that exactly two frames collided, the expected amount of peaks is therefore four.

```
% Find peaks above a parametrized threshold
lts_peaks = find(abs(correlation) > LTS_CORR_THRESHOLD*max(abs(correlation)));

% Assuming two frames, there will be four high peaks
[~, C] = kmeans(lts_peaks', 4);
uniq_lts_peaks = sort(floor(C))';

% Select the best candidate correlation peak at LTS-payload boundary
[LTS1, LTS2] = meshgrid(uniq_lts_peaks, uniq_lts_peaks);
[lts_second_peak_index, y] = find(iswithin( ...
    LTS2-LTS1, ...
    length(lts_t)/1.2, ...
    length(lts_t)*1.2 ...
));

% Add 128 samples for the symbol itself (at 40 MHz)
ind2.sig = uniq_lts_peaks(max(lts_second_peak_index)) + 128;
ind2.ltf = ind2.sig - 320; % subtract LTF length
ind2.stf = ind2.ltf - 320; % subtract STF length
ind2.payload = ind2.sig + 160; % add 4us SIG field
ind1.sig = uniq_lts_peaks(min(lts_second_peak_index)) + 128;
ind1.ltf = ind1.sig - 320;
ind1.stf = ind1.ltf - 320;
ind1.payload = ind1.sig + 160;
```

Listing 2: Collision Offset Detection using Meshgrid

The `meshgrid` function then calculates a two-dimensional table of all possible offsets between each two peaks. Using the `find` function, I look for offsets that equal the length of one training symbol with a 20 percent margin. This margin makes the algorithm a bit

5 https://www.mathworks.com/products/communications.html

more reliable against slight timing imprecision. The offset between the collided frames is the delay between these symbol matches.
Lastly, the signal field sample indices are calculated as the last symbol's correlation peak plus the length of that symbol, which is 128 samples at 40 MHz sampling rate. Start indices of the LTF, STF, and Data Field are deduced accordingly.

The indices are stored in a structure, which is later used to cut out the samples containing the sender's MAC address. That set is determined as the start of the address in the first frame until the end of the address in the second frame. Especially on real hardware, this allows to mitigate possible delays and offset between the frames.

5 EVALUATION

In this chapter, the proposed technique for detecting sender MAC addresses during collisions is evaluated using several simulations as well as experiments on Wireless Open-Access Research Platform (WARP) Software-Defined Radios (SDRs).
Possible questions and problems include the fact that with higher Modulation and Coding Schemes (MCSs), there might not be enough samples to correlate the rather short six byte MAC address. Another interesting issue is the accuracy of the algorithm with increasing amounts of noise and fading.

5.1 FREQUENCY-DOMAIN CORRELATION

During the implementation of the MAC address detection technique, I tried using cross-correlation in the frequency domain. This means that instead of calculating the similarity between samples, complex symbols are correlated without mapping them into the time domain. With this approach, the Fourier transform must be applied to the received signal.
It turns out to be very difficult, if not impossible, to detect senders in case of a collision with frequency-domain correlation. The reason for that is phase shift due to path delays. When a signal is received with an offset in the time domain, the constellation is changed in the frequency domain. This can be shown using the Fourier transform equation:

$$F(f) = \int_{-\infty}^{\infty} s(t) \cdot e^{-2\pi i f t} dt$$

Here, i is the imaginary unit, $F(f)$ is the Fourier transform at frequency f, and $s(t)$ is the time-domain signal depending on t. A time offset can now be expressed as $t + \Delta$. Standard power laws then show that this offset introduces a constant multiplication factor on $F(f)$, depending on Δ.

$$F(f) = \int_{-\infty}^{\infty} s(t+\Delta) \cdot e^{-2\pi i f \cdot (t+\Delta)} dt = \int_{-\infty}^{\infty} s(t+\Delta) \cdot e^{-2\pi i f t} \cdot e^{-2\pi i f \Delta} dt =$$
$$= e^{-2\pi i f \Delta} \cdot \int_{-\infty}^{\infty} s(t+\Delta) \cdot e^{-2\pi i f t} dt$$

Since both $s(t)$ and $F(f)$ are complex functions, this multiplication is a rotation, or phase shift, in the complex constellation plane. Furthermore, the rotation is different for diverse frequencies f.
Due to the way Orthogonal Frequency Division Multiplexing (OFDM) works, a received signal will contain a variety of frequencies within the channel bandwidth. All those frequencies experience a different phase shift. This leads to a received constellation similar to that shown in Figure 6. To create this figure, I applied a Rayleigh channel [25] to a simulated collision signal, introducing a delay of a few nanoseconds. The signals

are modulated with Binary Phase Shift Keying (BPSK), so ideally the only constellation points should be 1 and −1. However, due to the above described rotation, the constellation is basically a circle.

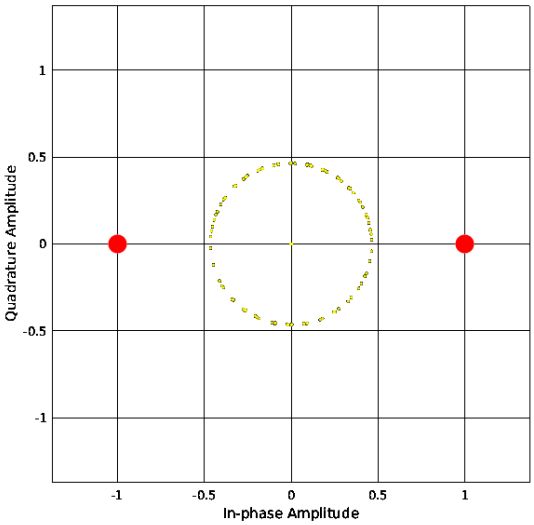


Figure 6: Cross-Correlation of Frequency-Domain Symbols after Channel

This kind of path effects is not at all unusual in a IEEE 802.11 transmission. Under normal circumstances, the known preamble is used to calculate the inverted channel matrix of the path effects [23], as mentioned in section 2.4.

When receiving a collision, there is however not only one preamble. Instead, the preambles of two different frames, that experienced different path effects, are added together. It is not easily possible to separate the two channel effects from each other.
This is caused by the receiver, which would normally correlate the received training field with a local copy, and estimate the phase shift present in the complex cross-correlation. For this reason, I can not reverse the phase shift introduced on the OFDM symbols containing the MAC address. It is therefore not possible to detect which sender transmitted the frame.
As expected, experiments with cross-correlation in the frequency domain without channel equalization were unsuccessful. I therefore disregarded this approach.

5.2 TIME-DOMAIN CORRELATION

The experiments I did using time-domain cross-correlation are divided into two groups. First, I used Matlab to simulate different field variations and channel effects. Second, I used WARP boards to try the approach in a real-world scenario. The results of these experiments are summarized in the following sections 5.4 to 5.8.
When correlating time-domain samples, I used pre-generated reference samples and calculated the correlation only for the subset of samples that contains the MAC address. Section 4.3 describes which period of the signal this is.

Reference signals are modulated until just after the Inverse Fast Fourier Transform (IFFT) and adding of cyclic prefix. For the simulations, I used a sampling rate of 20 MHz, which is the default output of the Matlab WLAN System Toolbox. The WARP boards use a sampling frequency of 40 MHz however, so for these experiments I used interpolation to double the rate.
All experiments used real-world MAC addresses, as described in section 5.3.

5.3 REAL-WORLD MAC ADDRESSES

MAC addresses are six byte long numbers, where the first three bytes are a vendor prefix, identifying the company that built the network interface. The remaining three bytes are a unique identifier for the specific hardware.
After initial tries with some artificial MAC addresses in the form of `AB:CD:EF:12:34:56`, I switched to using a sample set of real-world MAC addresses. This allowed for a closer simulation of an actual use-case for sender detection.

I collected 64 MAC addresses from devices that were connected to the wireless university eduroam network. The gathering was done in the afternoon on a weekday. I used airodump from the `aircrack-ng` software suite[1] to dump all MAC addresses on the network into a file. The actual command looked like this:

```
airmon-ng start wlp0s20u1
airodump-ng --essid eduroam -a -o csv -w mac-addresses-eduroam.csv wlp0s20u1mon
```

Listing 3: Capture Real-World MAC Addresses

It is important to have the wireless network interface set to promiscuous mode. This mode instructs the driver to hand all received frames to the kernel. Otherwise, only frames that are going to or coming from the local station are gathered, while the rest is filtered. This means that only the router's and the station's own MAC address would be cached.
In a possible real-time usage scenario of the sender detection algorithm, it would also be important to use promiscuous mode in order to collect all MAC addresses. These are the addresses for which reference signals need to be modulated and cached.

5.4 MODULATION AND CODING SCHEMES

Intuitively, the overall sender detection accuracy should decrease for higher MCSs. First, with a higher MCS there are more payload bits encoded in every OFDM symbol, as seen in Table 1. This means that when applying cross-correlation, the OFDM symbol contains random data from the MAC header before and after the sender MAC address. Due to the nature of OFDM, it is not possible to limit correlation to only a fraction of the symbol in the time domain.
Second, a higher MCS means that payload data gets mapped to complex points in the constellation plane that are less distinct from each other. Therefore, different data can be more similar under cross-correlation, making it harder to detect senders in the time

1 https://www.aircrack-ng.org/

domain.

To evaluate the detection algorithm's performance, I measured the correctness of detected MAC addresses for each of the eight available MCSs. For every MCS, 1000 runs were performed with a new randomly chosen sender pair on each run. There was no channel model, instead an ideal channel was assumed. The results are stacked sums of runs where a completely correct pair of senders were guessed, times where only one of the senders was correct, and the number of failures, in the sense of both addresses being incorrect.

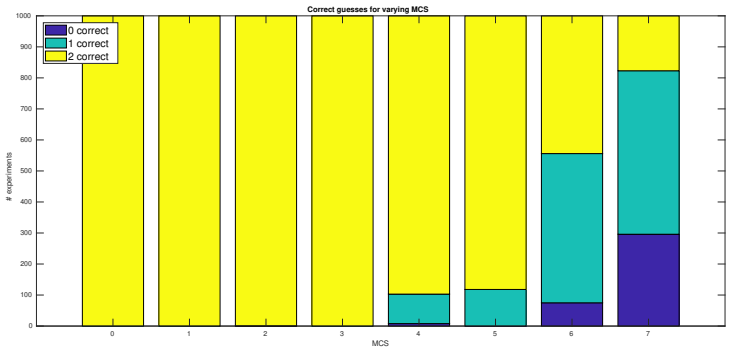


Figure 7: Results: Varying MCS for 1000 Runs

Figure 7 shows the resulting plot. Different MCSs are spread out on the x axis. The stacked bars show the number of experiments for each result category as described above. As expected, the detection accuracy decreases for higher MCSs. It is worth mentioning that for low MCSs, the performance is very good, better than I expected. Furthermore, it seems to decrease exponentially, but to justify this more data would be needed.

5.5 SCRAMBLER INITIALIZATION

As described in section 2.1, the MAC layer payload data is scrambled before any modulation on the physical layer is applied. The scrambling uses a seven bit state register, where 0 is no valid state. Therefore, 127 unique scrambler initialization values are possible.
If the scrambler initialization is important for the sender detection technique, specifically if the modulated reference signals must use the same initialization as the received frames, the number of possible values linearly scales the algorithm's complexity. That is, for every cached MAC address, MCS, et cetera all scrambler initializations must be modulated and correlated. It is therefore very interesting to know whether this is the case, or whether the scrambler does not affect detection quality.

Figure 8 shows the detection performance for different scrambler initializations over 1000 runs with an ideal channel. The data is presented the same way as in the previous section. The stacked bar plots denote the number of runs with zero, one, and two corrects guesses. For all experiments, the MCS 0 was used. While the simulated received

frame was modulated with scrambler initialization 1, correlation was calculated against 127 reference signals with different scrambler values.

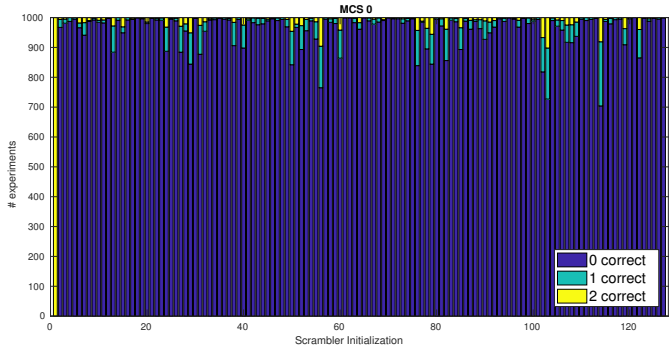


Figure 8: Results: Varying Scrambler Initialization for 1000 Runs at MCS 0

The results show that the scrambler initialization is very important for detection quality. While the performance is very good for the correct initialization value 1, for all other values the accuracy is low. However, related research mentions that some network interfaces seem to not choose the scrambler initialization at random [19], contradictory to the IEEE 802.11 standard. These devices use a static initialization for every frame, or increment the value after every transmission. This could be exploited by limiting the reference signals to the ones that are expected in the current network. This avoids time being spent on correlations that are unpromising. Since the MAC addresses of stations in the network are cached, the interface types and vendors are known. This could lead to some interesting future work.
Another important task is to identify when the correct scrambler initialization for a given frame is used and hence a sender guess is meaningful. Since the initialization values of collided frames are unknown, every possible value has to be tried and correlated. The sender guess is based on highest probability, meaning there is a guess regardless of whether a correlation is using the right scrambling sequence. Therefore, receivers need to detect which sender guess is the most promising one. This should be possible by using a threshold for cross-correlation peaks and could be explored in following work.

5.6 PRECEDING DATA VARIATIONS

Similar to the scrambler initialization, the MAC header field preceding the sender MAC address could also influence detection quality. This is due to the convolutional encoder as described in section 2.1. Since it uses a state register to calculate the encoding, the output bits are dependent on previous input. Therefore, it could be necessary to address all possible values of preceding data in order to generate suitable reference signals for all MAC addresses. The field before the sender MAC address in a data frame is the destination MAC address. In the following experiment, I measured to which extent the value of the destination address affects the accuracy of my detection technique. The scrambler initialization was 1 for all runs of this experiment. I used an ideal channel.

Figure 9 shows the results in the same way as for the previous figures. Since the convolutional encoder uses a seven bit state, only the last seven bits of the destination MAC

address matter. After these last seven bits, the register is synchronized to the same state, regardless of the preceding first 41 bits of the address. This makes up for a total of 128 different values that are evaluated with 1000 runs each. Unlike the scrambler initialization, results are presented for MCS 1 here. At MCS 0, there were no recognition errors at all, yielding a less interesting plot.

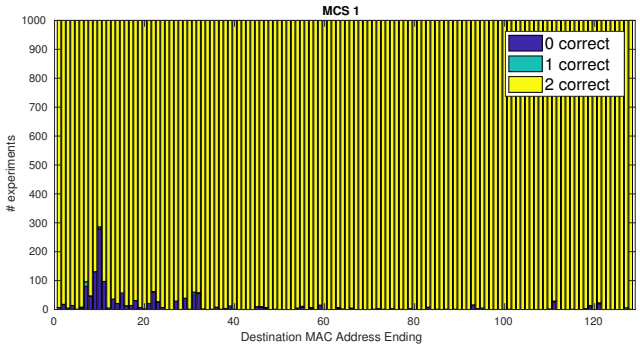


Figure 9: Results: Varying Destination MAC Address for 1000 Runs at MCS 1

It turns out that unlike the scrambler initialization, preceding payload data does not really have a measurable effect. Detection quality is comparable regardless of the trailing bits of the MAC address. I discuss possible reasons for this in section 6.2. An important implication of this result is that it is not necessary to check all 128 possible encoder states for every collision in real-time, which would be another linear scaling factor for the computational complexity of the algorithm.

5.7 CHANNEL MODELS

So far, all experiments were done in simulations with an ideal channel, meaning there were no path effects such as multi-path fading or attenuation. I evaluate performance under various channel models in this section.

Using Matlab, there are different models available that could be applied. First, I begin with applying Additive White Gaussian Noise (AWGN) with different Signal-to-Noise Ratios (SNRs). While AWGN is not specifically modeled for IEEE 802.11 networks and will therefore not replicate channel conditions precisely, it is a good starting point. The SNR is measured in decibel, and swept from -20 to 60dB.
The results of the experiments are shown in Figure 10. For each SNR and every MCS, 1000 experiment runs were performed.

Second, the Communications System Toolbox comes with `stdchan`, which supports the `802.11a` and `802.11g` channel types. These are suited quite well for this scenario. The channel models require some parameters, including the sampling frequency, Doppler spread, and Root Mean Square (RMS) delay spread t_{RMS}. The delay spread was varied from 100 ns to 500 ns in steps of 50 ns. The usage of the stdchan filter is illustrated by Listing 4:

```
trms = ...; % 30, 40, etc.
fs = 20e6; % Hz Sampling Frequency
```

```
fd = 10; % Hz Doppler spread
chan = stdchan(1/fs, fd, '802.11g', trms);
rx = filter(chan, tx);
```

Listing 4: Matlab *stdchan* Channel Model

The results of this experiment are shown in Figure 11. For each t_{RMS} value, 1000 runs were performed. The process was repeated for all eight MCSs.

Third, the WLAN System Toolbox comes with a number of specifically drafted IEEE 802.11 channel models. These are designed according to the IEEE 802.11 standard models described in the specification [1]. There are six models (A to F), of which I used the following three:

- *Model B*: typical large open space and office environments, no line-of-sight, 100 ns RMS delay spread
- *Model D*: large open space (indoor and outdoor), line-of-sight conditions, 140 ns RMS delay spread
- *Model E*: typical large open space (indoor and outdoor), no line-of-sight, 250 ns RMS delay spread

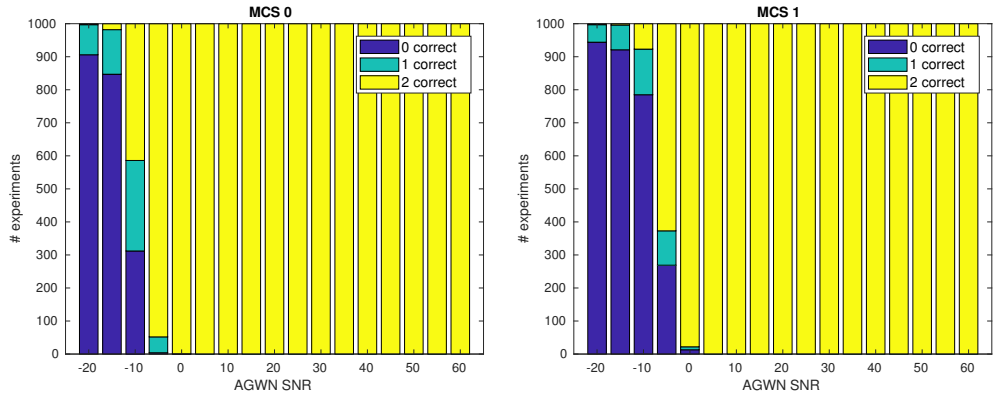
I used the `wlanTGnChannel` class. Technically, this channel is designed to be used with IEEE 802.11n high-throughput networks. However, the class supports configuration of Multiple Input, Multiple Output (MIMO) streams. I set those to one antenna for transmission and reception, respectively. This should provide appropriate conditions for Single Input, Single Output (SISO) 802.11a/g networks.
The channel is used with Matlab code similar to Listing 5. As for the `stdchan` invocation, the sampling rate is passed to the function. I use path-loss and shadowing for fading, and configure SISO.

```
model = ...; % 'Model-B', 'Model-D', etc.
tgnchan = wlanTGnChannel( ...
        'SampleRate', 20e6, ... % Hz
        'LargeScaleFadingEffect', 'Pathloss and shadowing', ...
        'NumTransmitAntennas', 1, 'NumReceiveAntennas', 1, ... % SISO
        'DelayProfile', model);
rx = tgnchan(tx);
```

Listing 5: Matlab *wlanTGnChannel* Simulation

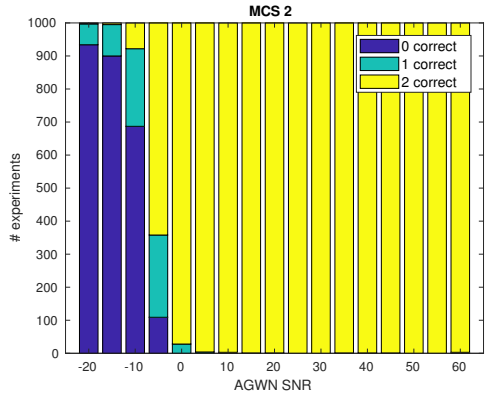
The results of the experiments are shown in Figure 12. For each of the three channel models and every MCS, 1000 runs were performed.

I conclude that MCSs 0 to 2 allow to detect senders quite well under channel effects. For these, AWGN caused almost no measurable impact. Even `stdchan`, which seems to have the biggest effect on detection quality, showed good results. For higher MCSs however, accuracy quickly decreased significantly. The combination of high RMS delay spread under the Standard Channel model with MCS starting from 3 yielded almost no entirely correct guesses.

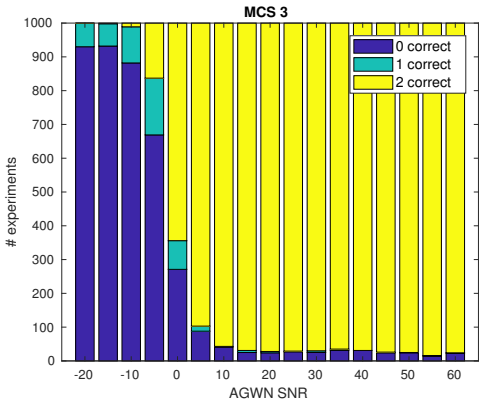


(a) MCS 0

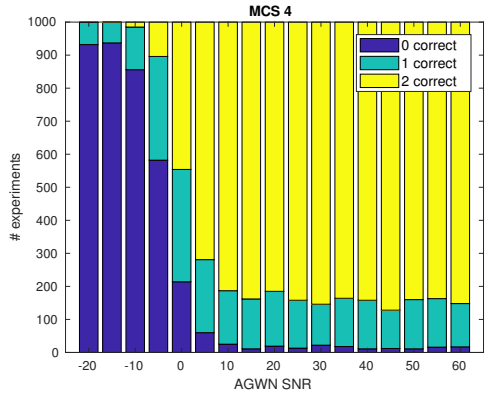
(b) MCS 1



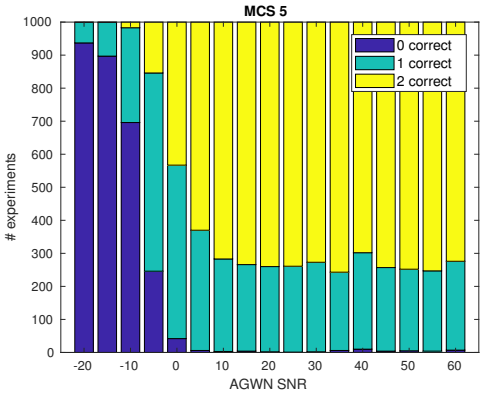
(c) MCS 2



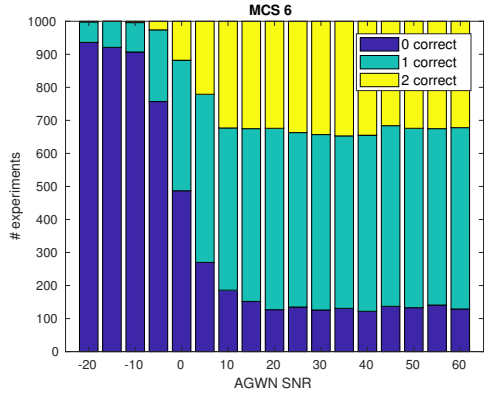
(d) MCS 3



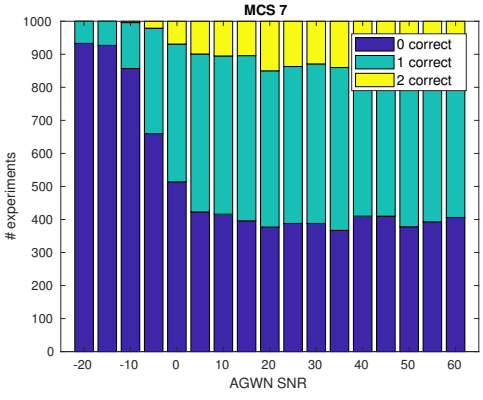
(e) MCS 4



(f) MCS 5

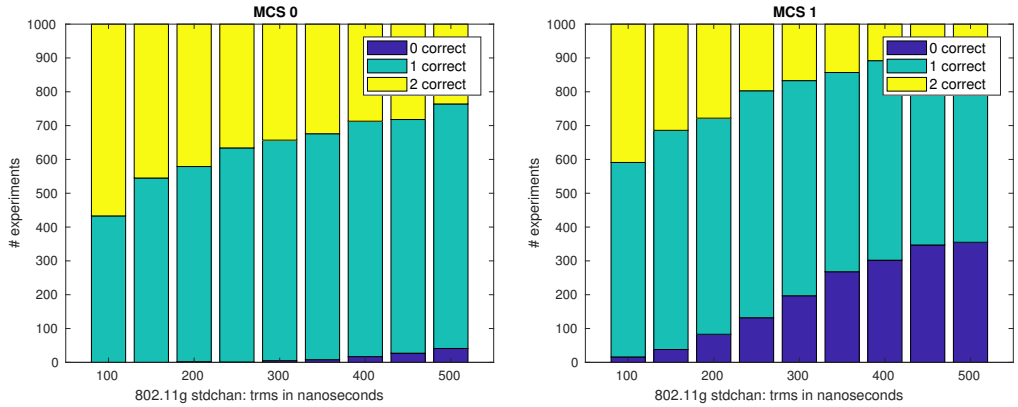


(g) MCS 6



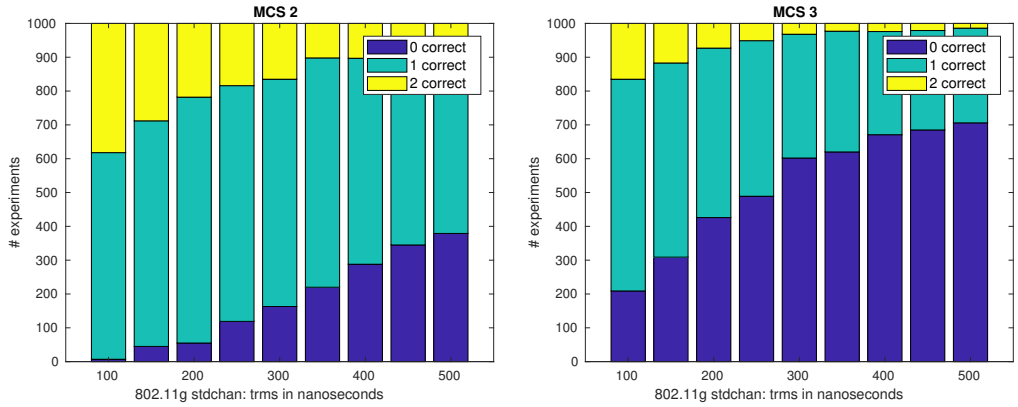
(h) MCS 7

Figure 10: Results: Varying AWGN SNR for 1000 Runs



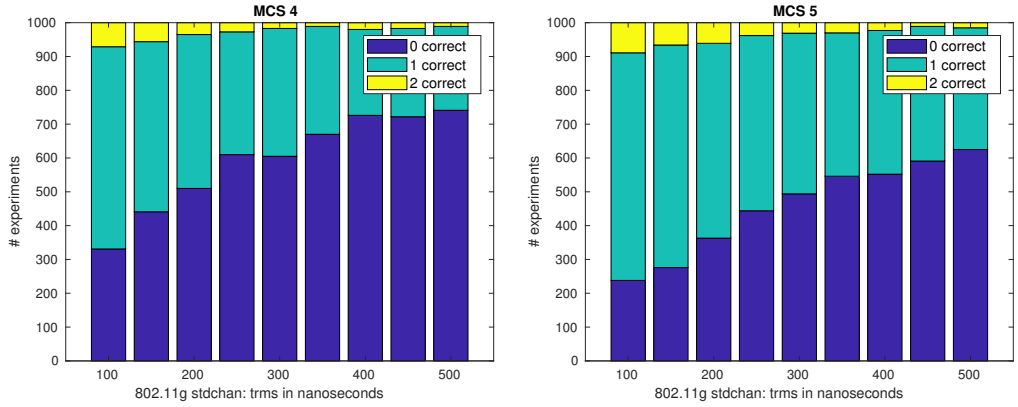
(a) MCS 0

(b) MCS 1



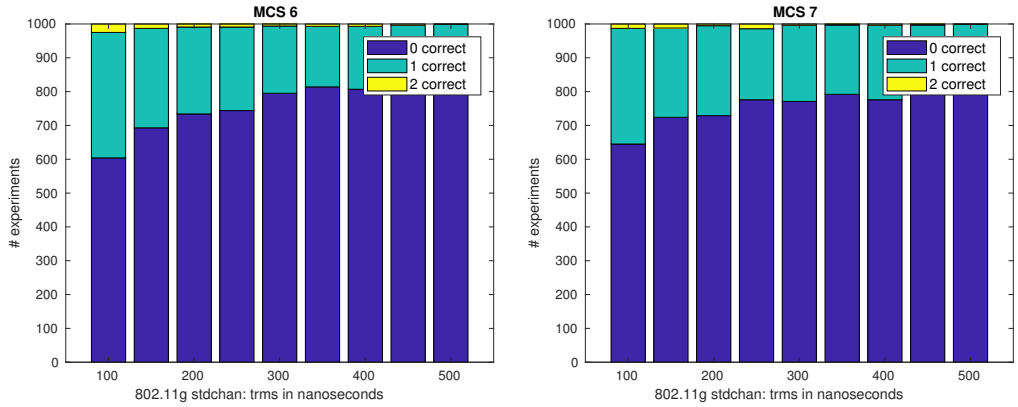
(c) MCS 2

(d) MCS 3



(e) MCS 4

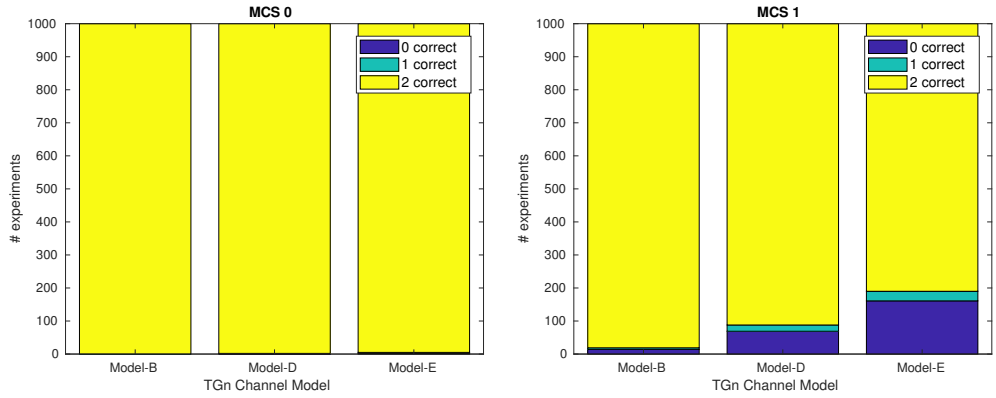
(f) MCS 5



(g) MCS 6

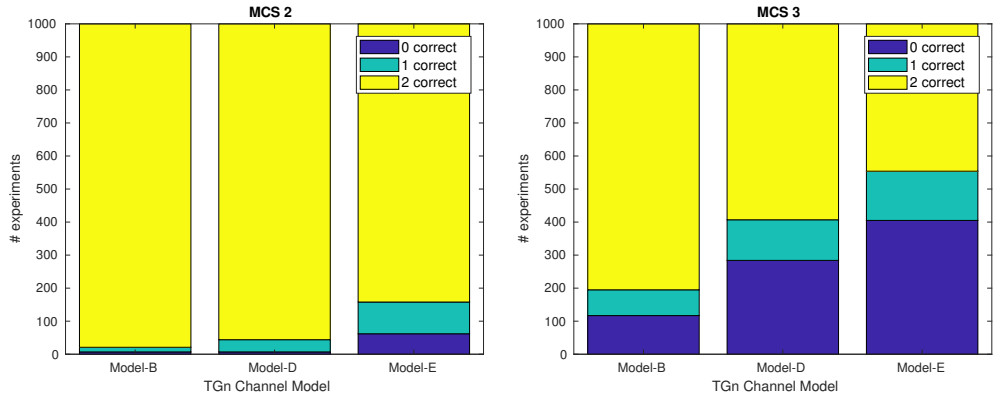
(h) MCS 7

Figure 11: Results: Varying t_{RMS} in a Standard Channel for 1000 Runs



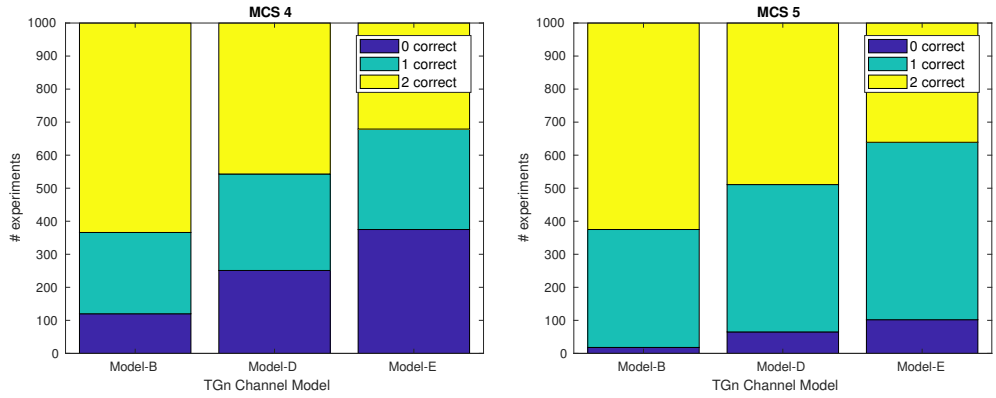
(a) MCS 0

(b) MCS 1



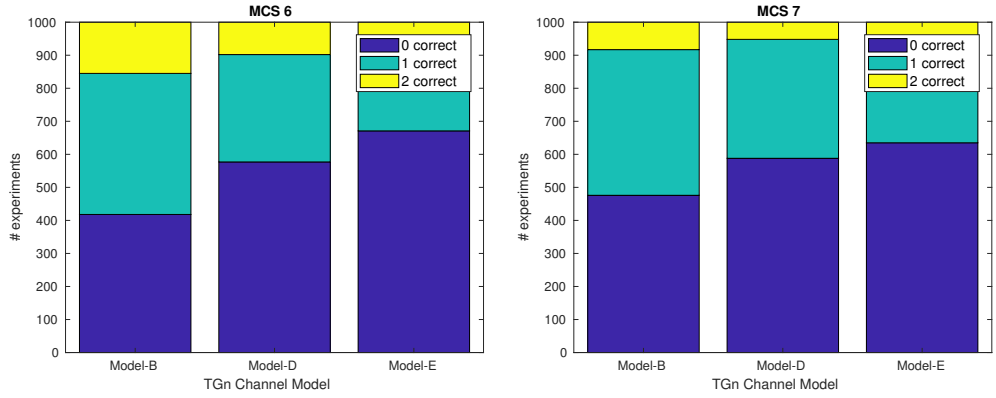
(c) MCS 2

(d) MCS 3



(e) MCS 4

(f) MCS 5



(g) MCS 6

(h) MCS 7

Figure 12: Results: Varying TGn Channel for 1000 Runs

5.8 WARP EXPERIMENTS

Lastly, I used SDRs instead of pure software simulations to evaluate the sender detection performance with real-world channel effects. The experiments were conducted using three WARP boards. I flashed all of them with the WARPLab reference design, version 7.5.1. Two SDRs were used as senders, the remaining one acted as receiver. The SDRs are connected to a controlling workstation using a gigabit Ethernet switch. The overall hardware layout is illustrated in Figure 13.

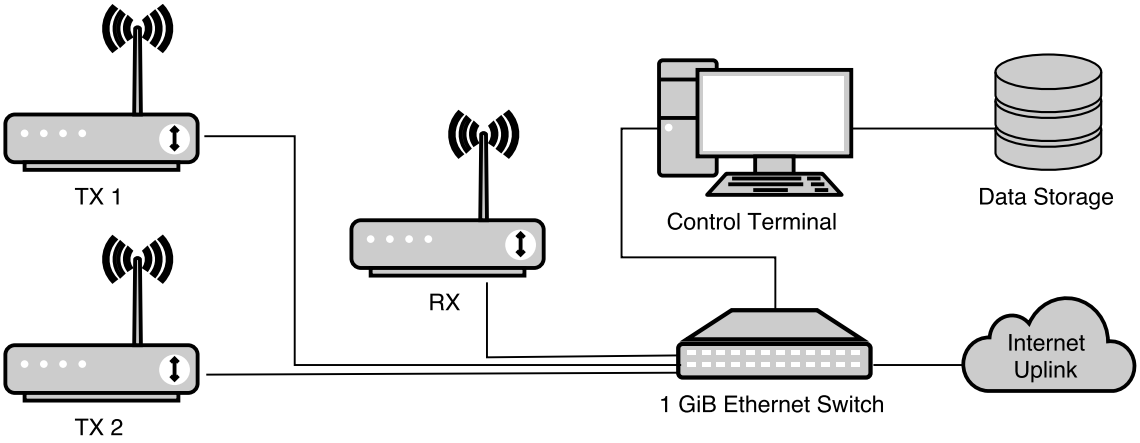


Figure 13: WARP SDR Experiments Hardware Layout

The experiments were carried out in a typical office building. This means that there were multiple IEEE 802.11 networks with several dozens of connected clients. I used the standard WLAN channel 11 and no electromagnetic shielding.

Since the WARP boards use a sampling frequency of 40 MHz, but the Matlab WLAN System Toolbox creates time-domain samples with a 20 MHz sampling rate, I used interpolation to adapt frequencies. This is done with the built-in `resample` Matlab function. Listing 6 illustrates the process.

```
% Interpolate to get from 20 to 40 MHz sampling rate
tx1 = resample(tx1, 40, 20);
```

Listing 6: Interpolate Sampling Rate

For the real-world experiments, I deliberately caused frame collisions with the same scrambler initialization and destination MAC addresses. This means that in principal everything was the same as for the most basic simulation experiments.
To assess the basic mechanics of receiving collisions with the SDRs, I applied cross-correlation to only one Long Training Field (LTF) symbol instead of the MAC addresses and plotted the results. They are presented in Figure 14. The collided frames were aligned with some offset for this experiment.
While one group of spikes is slightly lower than the other, it is still easily possible to detect that there is a collision due to the four higher and two lower spikes caused by the two LTFs. The higher level of noise in the beginning of the signal are probably caused by Automatic Gain Control (AGC) in the receiving SDR. Since the MAC addresses are

located further into the signal, and only that part is correlated for sender detection, this noise should be no problem.

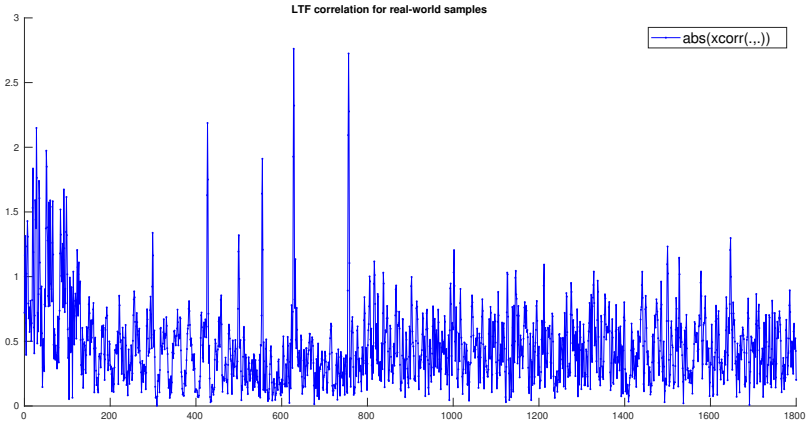


Figure 14: Preamble Correlation on WARP SDRs

The results of the preamble correlation are promising. They show that indeed a collision of two frames is captured by the receiving SDR, and that cross-correlation in the time domain can in fact distinguish both LTFs.

Next, I used the same code as for the simulation experiments to try and detect sender MAC addresses. This worked reasonably well despite some problems. The results are shown in Figure 15. I used MCSs 0 to 5 with a total of 10 runs each.

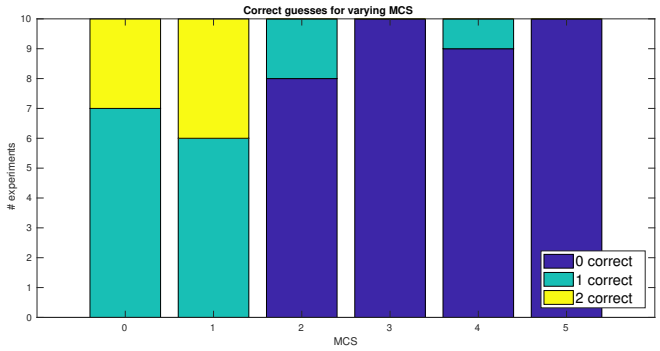


Figure 15: Results: Varying MCS for 10 Runs on WARP SDRs

For MCSs 0 and 1, the results were quite good. Although in most of the cases only one MAC address could be detected correctly, investigations showed that the correctly guessed address was transmitted by the same SDR every time. This hints that there could be some environmental factors causing its signal to be received more strongly. Another thing is that wrong guesses were often due to quite similar MAC addresses, which were different in only a few bits.
However, for higher MCSs, sender detection seemed to fail. This is probably caused by real-world channel effects, that could not be simulated to a realistic extent.

6 DISCUSSION

This chapter discusses the results of the evaluation, and their implications. Finally, future work is proposed to iterate on the findings.

6.1 COMPUTATIONAL COMPLEXITY

In order to use the proposed sender detection algorithm, it must be feasible to do all necessary calculations in real-time. In this section, I present an analysis of the computational complexity of the different components.
Two main categories of tasks can be identified. First, the generation of reference signals to use for cross-correlation. This can be pre-computed before any collisions are received and is therefore less time-critical. Second, correlating time-domain samples with the reference signals and sorting to work out a guess which senders participated. This needs to be done for every analyzed collision.

Based on the results from the experiments with scrambler initialization and the destination MAC address as described in sections 5.5 and 5.6, the following factors scale up the amount of necessary reference signals:

- MAC addresses on the network
- Modulation and Coding Schemes (MCSs)
- Scrambler initialization values

The destination MAC address can be ignored. This means that the algorithm scales according to the following equation:

$$N_{RS} = N_{MAC} \cdot N_{MCS} \cdot N_{SI} = N_{MAC} \cdot 8 \cdot 127$$

Here, N_{RS} denotes the number of modulated reference signals, N_{MCS} is the number of available MCSs, and N_{SI} describes the amount of possible scrambler initialization values. Let $n = N_{MAC}$ be the number of cached MAC addresses on the network. The worst-case asymmetric complexity in this case is:

$$O(\text{"Sender Detection"}) = O(n \cdot 8 \cdot 127) = O(1016n) = O(n)$$

The algorithm scales linearly with the amount of stations on the IEEE 802.11 network. This is optimal since every station is a possible sender which has to be considered when decoding a collision. It is however debatable whether the linear factor of 1016 is feasible on commodity computing hardware. That would be necessary to enable broad usage of the detection technique across many different devices as mentioned in chapter 1. A

quantitative analysis is therefore desirable.

I used the Matlab profiler to gather data on the execution speeds of different parts of the algorithm. These measurements were done on a consumer laptop, featuring a 2-core 3rd-generation Intel i3 processor with Hyper-Threading at 1.9 GHz. The results are summarized in Table 3.

Function	**Time Spent**
wlanTGnChannel	912 ms
wlanNonHTData	662 ms
CRC Calculation	122 ms
awgn	54 ms
xcorr	< 1 ms

Table 3: Timing Analysis of the Detecting Algorithm

This data suggests that the real-time part of the algorithm, which is limited to cross-correlation with `xcorr`, is negligible compared to the pre-modulation of reference signals. For small networks, the correlation time is probably fast enough. For a high number of stations however, the complexity gets out of hand quite quickly. For a network with 500 clients for example, each collision requires a calculation of about 500 seconds due to the linear scaling factor of 1016. However, it is easily possible to parallelize this workload, or even use specialized hardware such as Field Programmable Gate Arrays (FPGAs). In the extreme case, where every correlation is done at the same time, only 1 ms is needed to decode the collision.
The calculation of Cyclic Redundancy Check (CRC) checksums is time-consuming, yet not required for the algorithm. Instead, dummy checksums can be used as described in section 4.4. Therefore, modulation of reference signals scales with the performance of the `wlanNonHTData` function.
Since the modulation of reference signals can be done ahead of time, the only important case is when a new client connects. This requires 1016 new signals to be created for the new MAC address, which takes about five minutes without any parallelization. With more CPU cores however, it should be possible to reduce the time to an acceptable limit.

6.2 DETECTION QUALITY

The results of my simulations are promising. While admittedly only IEEE 802.11a/g networks were evaluated, as section 6.3 discusses further, detection accuracy was quite good even for higher MCSs.

In contrast to the scrambler initialization, the value of the preceding destination MAC address had hardly any effect on detection quality, as described in section 5.6. Most likely this is due to the relatively small state of the convolutional encoder compared to the size of a MAC address. While the encoder considers the last seven seen bits for its output, a MAC address contains 48 bits of data.

Regardless of the preceding destination MAC address, the convolutional encoder state is synchronized after the first seven bits of the sender address. This is only a small fraction of about 15 %. In addition to that, the first seven bits are part of the vendor prefix, which is already to some extent likely to be the same across multiple stations on the network. Therefore, convolutional encoding poses no critical impact on sender detection performance.

Using real-world Software-Defined Radios (SDRs), the detection technique performed reasonably well at least for lower MCSs. This is a particularly nice result as these experiments were conducted in an environment where regular wireless traffic was present. Although there was some decrease in detection quality, this shows that the algorithm is capable of successfully recognizing sender MAC addresses even when a reasonably high network load is present.

6.3 IEEE 802.11N/AC/AX NETWORKS

This thesis only covered sender detection for IEEE 802.11a/g networks. However, such networks are rarely used nowadays due to their low throughput. Modern standards such as IEEE 802.11n/ac, and the upcoming 802.11ax are much more relevant.
There are many improvements and differences introduced with the 802.11n standard. One of the most important ones with respect to time-domain sample cross-correlation is the adoption of Multiple Input, Multiple Output (MIMO) transmission. With this technique, every sender and receiver uses multiple antennas, instead of just one as with 802.11a/g. This allows for a much higher spectral efficiency, meaning that more bits can be transmitted per used bandwidth [23]. However, the overall system complexity and multi-path effects in particular make it much more difficult to apply naive time-domain correlation.

It would be interesting whether the here proposed sender detection algorithm can be adapted to work in a MIMO environment. In the current form, this is unfortunately quite unlikely. On the one hand, collision detection and especially the determination of relevant sample periods containing the sender MAC addresses must be adjusted to the physical layer high-throughput frame format used by modern standards [1].
On the other hand, using multiple streams implies the possibility that the sender's MAC address gets fragmented. While some bits can be transmitted in the first stream, the remaining bits could be sent in various other streams. This renders simple cross-correlation with `xcorr` directly on the received sample vector ineffective.

6.4 FUTURE WORK

The preceding discussion reveals some serious problems with the proposed algorithm for sender MAC address detection. The fundamental principles however look promising. There are remaining questions that should be addressed in further research. Some of them are presented in this section.

Quantitative Noise Evaluation with SDRs

As mentioned in section 6.2, noise from other stations in the network could cause the correlation of MAC addresses to degrade. Whether this is the case, and to which extent it influences detection performance, could be evaluated in a quantitative analysis. As a first step, the experiments with Wireless Open-Access Research Platform (WARP) SDRs should be reproduced in a radiation-free environment such as a Faraday cage. If the technique performs better in that case, additional IEEE 802.11 devices can be added one at a time, while continuously measuring the algorithm's accuracy.

MIMO and IEEE 802.11n/ac/ax

In a further step, the technique could be adapted to current IEEE 802.11 standards, such as 802.11n/ac. This involves finding a solution to the problems with MIMO, as described in section 6.3, especially the fragmentation of MAC address bits onto different spatial transmission streams.

Implementation on Mobile Operating Systems

In order to be used with possible new Distributed Coordination Functions (DCFs), it is interesting how the sender detection algorithm performs on mobile operating systems, Android and Apple iOS in particular. With the increasing amount of smartphone usage nowadays, a significant percentage of stations in IEEE 802.11 networks are based on these operating systems. While desktop operating systems are also very important, mobile platforms face the additional issue that due to limited battery power, calculations are inherently more expensive.
Future work could implement sender detection on collisions as a kernel module for mobile platforms and evaluate its performance and accuracy. The Nexmon project [1] could be a great framework for this. A special focus should be set to complexity, power usage, and overall implications on device usability and responsiveness.

Generalization of the Approach

I focused on collisions between two stations in this thesis. However, the described detection technique should be applicable to the general case. As more senders collide, more Long Training Field (LTF) correlation spikes are captured. This allows to count how many stations are involved in a collision. A receiver can then take the appropriate number of MAC addresses sorted by their correlation peaks to guess all senders.
Furthermore, instead of simply guessing the MAC addresses with highest correlation, future work could introduce an algorithm that uses the exact correlation values and ratios. This could maybe be combined with a learning-based approach to prefer the most useful features based on the current environment and conditions.

1 https://nexmon.org

7 CONCLUSION

Recognizing senders by correlating their MAC addresses in an IEEE 802.11a/g frame collision is an approach that to my knowledge has not been explored before. In this thesis, I introduced an algorithm to apply this technique and developed a proof-of-concept implementation using Matlab with the Communications and WLAN System Toolboxes.

The technique promiscuously listens on the wireless network interface to cache MAC addresses of stations connected to the network. For each of these addresses, several reference signals are pre-modulated and stored, considering different Modulation and Coding Schemes (MCSs) and scrambler initialization values. When a collision is received, samples containing the MAC address are cross-correlated in the time domain. The reference signals with the highest correlations are used to determine the senders participating in the collided transmission. The proposed technique was evaluated in simulations and in practical testbed experiments with 64 real, captured MAC addresses from the university eduroam network.

In simulations, the proposed technique performed very well even for higher MCSs. Varying the scrambler initialization caused severe regression, whereas the destination MAC address preceding the sender played a minor role. Simulations with standard channel models showed measurable impact, however up to a certain point sender detection retained a reasonable accuracy.

Experiments with real hardware, on three Wireless Open-Access Research Platform (WARP) Software-Defined Radios (SDRs) in particular, showed promising results. Receiving a collision, correlating the frame preambles, and measuring a delay between two collided frames worked very well. Detecting sender MAC addresses at MCSs 0 and 1 was similar to simulations and provided reasonable sender guesses. Due to channel effects, the accuracy was however reduced. The algorithm produced no usable results for higher MCSs.

The proposed algorithm is not directly applicable to current versions of the IEEE 802.11 standard using Multiple Input, Multiple Output (MIMO). This is due to possible spatial fractioning of the time-domain samples containing the sender MAC address, rendering naive cross-correlation ineffective.

Future work could quantize channel effects on real hardware, and adapt the detection technique to modern IEEE 802.11n/ac/ax standards and MIMO transmissions.
In summary, sender detection by using cross-correlation in the time domain worked surprisingly well despite some problems.

BIBLIOGRAPHY

[1] Standards I. E. E. E. Association. *802.11-2012 - 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*. Tech. rep. IEEE Std 802.11™-2012. IEEE-Inst, 2012.

[2] Standards I. E. E. E. Association. *IEEE 802.3 Standard*. Tech. rep. IEEE-Inst, 2015.

[3] Tarun Bansal, Bo Chen, Kannan Srinivasan, and Prasun Sinha. "Mozart: Orchestrating collisions in wireless networks." In: *IEEE INFOCOM*. 2013, pp. 351–362.

[4] G. Bianchi. "Performance analysis of the IEEE 802.11 distributed coordination function." In: *IEEE Journal on Selected Areas in Communications* 18.3 (2000), pp. 535–547.

[5] Jun Cao, Jieun Yu, Kyunghwi Kim, and Wonjun Lee. "Behavioral learning of exposed terminals in IEEE 802.11 wireless networks." In: *2009 First International Conference on Ubiquitous and Future Networks*. 2009, pp. 208–213.

[6] H. H. Choi, H. Lee, and I. H. Lee. "Carrier sensing multiple access with collision resolution (CSMA/CR) protocol for next-generation wireless LAN." In: *2013 Fifth International Conference on Ubiquitous and Future Networks (ICUFN)*. 2013, pp. 220–225.

[7] Hermanto Chua and Ji-Hoon Yun. "Classification of Transmission Events Based on Receive Power Pattern with Self-Tuning Thresholds in Wireless Receivers." In: *Wireless Personal Communications* 90.3 (2016), pp. 1487–1495.

[8] Shyamnath Gollakota and Dina Katabi. "Zigzag Decoding: Combating Hidden Terminals in Wireless Networks." In: *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*. SIGCOMM '08. Seattle, WA, USA: ACM, 2008, pp. 159–170.

[9] Aditya Gudipati and Sachin Katti. "Strider: Automatic Rate Adaptation and Collision Handling." In: *Proceedings of the ACM SIGCOMM 2011 Conference*. SIGCOMM '11. Toronto, Ontario, Canada: ACM, 2011, pp. 158–169.

[10] Aditya Gudipati, Stephanie Pereira, and Sachin Katti. "AutoMAC: Rateless Wireless Concurrent Medium Access." In: *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*. Mobicom '12. Istanbul, Turkey: ACM, 2012, pp. 5–16.

[11] Daniel Halperin, M Josie Ammer, Thomas E Anderson, and David Wetherall. "Interference Cancellation: Better Receivers for a New Wireless MAC." In: *HotNets*. 2007.

[12] S. A. Hejazi and B. Liang. "Throughput Analysis of Multiple Access Relay Channel under Collision Model." In: *2010 Proceedings IEEE INFOCOM*. 2010, pp. 1–9.

[13] M. G. Jibukumar, S. Shajahan, P. Preetha, and G. Sethulekshmi. "Impact of capture effect on receiver initiated collision detection with sequential resolution in WLAN." In: *2015 International Computer Science and Engineering Conference (ICSEC)*. 2015, pp. 1–6.

[14] Sam M. Keene and Jeffrey B. Carruthers. "Collision Localization for IEEE 802.11 Wireless LANs." In: *Wireless Personal Communications* 63.1 (2012), pp. 45–63.

[15] Jae Hyun Kim and Jong Kyu Lee. "Capture effects of wireless CSMA/CA protocols in Rayleigh and shadow fading channels." In: *IEEE Transactions on Vehicular Technology* 48.4 (1999), pp. 1277–1286.

[16] Hong Lin and David McDonald. "Lock step: an algorithm to reduce wi-fi jitter." In: *IEEE Communications Letters* 13.7 (2009).

[17] S. Lv, X. Dong, Y. Lu, X. Du, X. Wang, Y. Dou, and X. Zhou. "3D pipeline contention: Asymmetric full duplex in wireless networks." In: *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. 2014, pp. 790–798.

[18] Tong Meng, Fan Wu, Aijing Li, Guihai Chen, and Nitin H. Vaidya. "On Robust Neighbor Discovery in Mobile Wireless Networks." In: *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*. CoNEXT '15. Heidelberg, Germany: ACM, 2015, 38:1–38:13.

[19] Guevara Noubir, Muriel Medard, and Peter Chin. "Wi(Deband)-Fi: A Proposal for an Opportunistic Wideband Architecture Based on Wi-Fi." In: *Proceedings of the 14th ACM International Symposium on Mobility Management and Wireless Access*. MobiWac '16. Malta, Malta: ACM, 2016, pp. 115–122.

[20] George Nychis, Thibaud Hottelier, Zhuocheng Yang, Srinivasan Seshan, and Peter Steenkiste. "Enabling MAC Protocol Implementations on Software-defined Radios." In: *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*. NSDI'09. Boston, Massachusetts: USENIX Association, 2009, pp. 91–105.

[21] J. C. Park, S. K. Kasera, and N. Patwari. "Cross Layer Multirate Adaptation Using Physical Capture." In: *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*. 2009, pp. 1–8.

[22] P. Patel and J. Holtzman. "Analysis of a simple successive interference cancellation scheme in a DS/CDMA system." In: *IEEE Journal on Selected Areas in Communications* 12.5 (1994), pp. 796–807.

[23] Eldad Perahia and Robert Stacey. *Next Generation Wireless LANs: 802.11N and 802.11Ac*. 2nd. New York, NY, USA: Cambridge University Press, 2013.

[24] L. R. Rabiner, B. Gold, and C. K. Yuen. "Theory and Application of Digital Signal Processing." In: *IEEE Transactions on Systems, Man, and Cybernetics* 8.2 (1978), pp. 146–146.

[25] B. Sklar. "Rayleigh fading channels in mobile digital communication systems. I. Characterization." In: *IEEE Communications Magazine* 35.9 (1997), pp. 136–146.

[26] T. Vermeulen, F. Rosas, M. Verhelst, and S. Pollin. "Performance analysis of in-band full duplex collision and interference detection in dense networks." In: *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*. 2016, pp. 595–601.

[27] Pei Wang, Qunfeng Dong, Mingjun Xiao, and Liusheng Huang. "Efficient Wireless Broadcasting Using Onion Decoding." In: *Wireless Algorithms, Systems, and Applications: 5th International Conference, WASA 2010, Beijing, China, August 15-17, 2010. Proceedings*. Ed. by Gopal Pandurangan, V. S. Anil Kumar, Gu Ming, Yunhao Liu, and Yingshu Li. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 225–234.

[28] Y. Wu, G. Zhou, and J. A. Stankovic. "ACR: Active Collision Recovery in Dense Wireless Sensor Networks." In: *2010 Proceedings IEEE INFOCOM*. 2010, pp. 1–9.

[29] X. Zhang and K. G. Shin. "Chorus: Collision Resolution for Efficient Wireless Broadcast." In: *2010 Proceedings IEEE INFOCOM*. 2010, pp. 1–9.

[30] Jumin Zhao, Shaolei Fan, Deng-ao Li, and Baofeng Zhao. "Collision Tolerance: Improving Channel Utilization with Correlatable Symbol Sequences in Wireless Networks." In: *Int. J. Distrib. Sen. Netw.* 2015 (Jan. 2015), 11:11–11:11.

[31] Y. Zhu and Y. Sun. "Packet-Level Failure Classification by Characterizing Failure Patterns in Wireless Sensor Networks." In: *2015 IEEE Global Communications Conference (GLOBECOM)*. 2015, pp. 1–6.

ERKLÄRUNG ZUR ABSCHLUSSARBEIT

gemäß § 22 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Johannes Tobias Lauinger, die vorliegende Bachelor Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.
Mir ist bekannt, dass im Falle eines Plagiats (§38 Abs.2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.
Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung überein.

Darmstadt, 10. August 2017

Johannes Tobias Lauinger