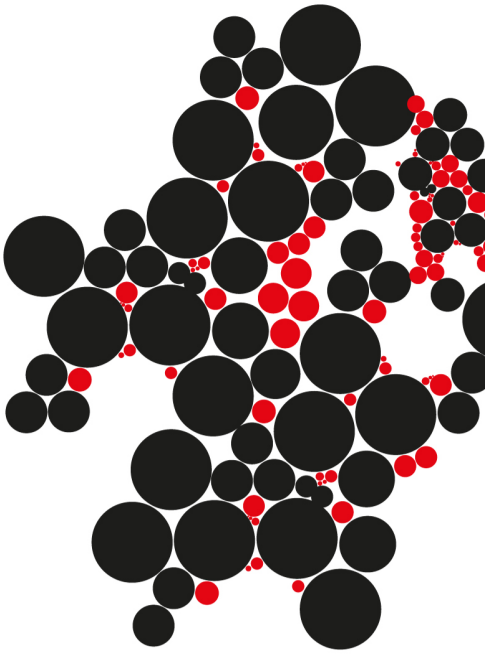
MASTER THESIS



FRAME CAPTURE IN IEEE 802.11P VEHICULAR NETWORKS

A SIMULATION-BASED APPROACH

P. van Wijngaarden, B.Sc

FACULTY OF ELECTRICAL ENGINEERING, MATHEMATICS AND COMPUTER SCIENCE

DESIGN AND ANALYSIS OF COMMUNICATION SYSTEMS

EXAMINATION COMMITTEE
dr. ir. G. Heijenk
dr. ir. G. Karagiannis
E.M. van Eenennaam, M.Sc.

UNIVERSITY OF TWENTE.

JULY 2011

Abstract

This thesis describes the occurrence of Frame Capture in the IEEE 802.11p physical layer. Frame Capture occurs when two nodes simultaneously transmit a message (a PLCP frame) that spatially and temporally overlaps at a certain receiver, creating a collision. Collisions can occur because of hidden terminals, or because of simultaneously ending backoff counters between coordinated nodes. If both frames are equal in signal strength, they both interfere in such a degree with each other that the receiver will not be able to decode either one of them. However, if one frame is received with a significantly higher power level than the other, the receiver can (under some conditions) be able to suppress the weaker frame and correctly receive the stronger one.

We studied the available literature on Frame Capture and concluded that the capture behavior is different among various chipsets and depends largely on the arrival time difference (in the order of microseconds) between both frames and on whether both frames interfere in each other's PLCP preamble or not. The literature compares Atheros and Prism chipsets, and demonstrates that if the stronger frame arrives before the weaker frame, both chipset types are able to capture the frame if the signal to noise ratio is high enough. If the stronger frame arrives later, the required SNR is significantly higher because the receiver must be able to clearly hear the stronger frame's preamble, even in the presence of interference, to lock onto the new frame correctly.

We implemented this capture behavior in our wireless network simulator, OMNeT++ with the MiXiM module, with the use of capture thresholds. Depending on the moment of arrival and the current state of the receiver, a certain threshold must be exceeded for the receiver to capture the frame. These capture thresholds are based on the results in the literature, but could be refined later on with experiments of our own. Note that capturing the frame (which happens after the preamble) does not guarantee correct reception: if for example during the frame more interference arrives, the frame can still be lost.

We also enhanced the bit error calculation & prediction formulas in MiXiM, which where only suited for 802.11b, a completely different physical layer. We now approximate bit errors with theoretical formulas derived for an AWGN channel (i.e. where white noise is the only interference). This is not perfect yet but a significant improvement over the current implementation, and it will help further increase the channel model accuracy in the future.

After the implementation work was complete, we performed simulations of vehicular networks with which we demonstrate that Frame Capture is especially important in vehicular networks with broadcast traffic. This is because the hidden terminal problem becomes more prevalent when all frames are broadcast, because the standard RTS/CTS mechanism can not be used to warn hidden terminals about ongoing transmissions. We demonstrate that both Atheros and Prism chipsets show a performance gain of at least 20% over (reference) chipsets which do not perform any kind of Frame Capture. By artificially disabling the hidden terminal problem (i.e. silencing all hidden terminals during a transmission) we show that Frame Capture is mostly important when resolving hidden terminal collisions. Collisions between coordinated nodes are more likely to have an equal power level and can usually not be resolved with Frame Capture. Because nodes do not update their MAC contention window if broadcast traffic is the only type of traffic in the network, occurrence of collisions between coordinated nodes increases with increased node density.

Samenvatting

Dit onderzoekt beschrijft Frame Capture in de fysieke laag van IEEE 802.11p. Frame Capture treedt op wanneer twee nodes tegelijkertijd een bericht (een PLCP frame) sturen naar een derde node. Dit creëert een *collision*, ofwel een botsing tussen twee frames. Collisions kunnen optreden door hidden terminals (als de twee zendende nodes elkaar niet kunnen horen), of door backoff timers (van de IEEE 802.11 MAC laag) die tegelijkertijd aflopen, waardoor ze op precies hetzelfde moment beginnen te zenden. Als de frames ongeveer dezelfde signaalsterkte hebben zijn beide transmissies doorgaans verloren gegaan, maar dit hoeft niet altijd het geval te zijn; als één van de twee een signaalsterkte heeft die significant sterker is dan de andere, kan de ontvanger soms toch het sterkere frame succesvol ontvangen. Dit wordt 'Frame Capture' genoemd.

We hebben de beschikbare literatuur over Frame Capture bestudeerd en geconcludeerd dat het capture gedrag verschilt tussen de chipsets van verschillende fabrikanten. Deze verschillen kunnen hun effect hebben op de performance van het gehele netwerk. Of een ontvanger in staat is om een frame te capturen hangt af van de signaal-ruisverhouding en het verschil in aankomsttijd (in de orde van microseconden); als de ontvanger (veel) interferentie heeft tijdens de preamble, wordt de kans op succes kleiner. Op dit punt verschillen Atheros en Prism chipsets ook van elkaar: Prism chipsets zijn alleen maar in staat om een sterker frame te capturen als het eerder aankomt dan het zwakkere frame, terwijl Atheros chipsets ook een sterker frame kunnen capturen terwijl de ontvanger al begonnen is met het ontvangen van het zwakkere frame. Afhankelijk van de status van de ontvanger en of de ontvanger nog meer interferentie heeft of niet, is de vereiste SNR (signal-to-noise ratio of signaal-ruis verhouding) hoger resp. lager.

We hebben dit capture gedrag geimplementeerd in onze netwerksimulator, het OMNeT++ framework met de MiXiM module. We gebruiken capture thresholds, dit zijn SNR waarden die een frame moet hebben om in het geval van een collision gecaptured te kunnen worden. Deze thresholds zijn gebaseerd op de resultaten van experimenten uit de literatuur, maar zouden later verbeterd of gevalideerd kunnen worden met eigen experimenten met IEEE 802.11p hardware. Let op dat het capturen van een frame niet automatisch leidt tot succesvolle ontvangst: afhankelijk van de hoeveelheid interferentie die de ontvanger heeft tijdens het data-gedeelte van het frame zou het frame nog steeds verloren kunnen gaan.

We hebben ook de bit error berekenings- & voorspellings-formules verbeterd in MiXiM.

Deze waren eerst gebaseerd op 802.11b, wat een geheel andere fysieke laag is (gebaseerd op DSSS). We benaderen bitfouten nu met behulp van theoretische formules die bit- en pakketfouten benaderen voor een AWGN-kanaal (met alleen witte ruis als interferentiebron). Dit is niet perfect en er zijn in het echt veel meer effecten die een rol spelen, maar deze benadering is in ieder geval een verbetering ten opzichte van de oude situatie en een stap voorwaarts richting een beter model voor het kanaal in de toekomst.

Nadat de implementatie voltooid en gevalideerd was, hebben we simulaties van autonetwerken uitgevoerd die demonstreren dat Frame Capture inderdaad een belangrijke positieve bijdrage levert aan de performance van een netwerk met hoofdzakelijk broadcast-verkeer. Dit komt doordat het hidden terminal probleem een grotere rol speelt in autonetwerken, omdat het standaard RTS/CTS mechanisme dat gebruikt wordt om collisions te voorkomen niet gebruikt kan worden bij broadcasts.

We hebben laten zien dat zowel Atheros als Prism chipsets minimaal 20% beter presteren dan een gewone chipset (ter referentie) met geen enkele vorm van Frame Capture. Ook presteren Atheros chipsets altijd tussen de 5 en 8% beter dan de Prism chipsets, dankzij het feit dat ze op elk willekeurig moment een frame kunnen capturen, niet alleen voor en tijdens de preamble van het zwakkere frame.

Door kunstmatig het hidden terminal probleem uit te schakelen (d.w.z. door te zorgen dat alle hidden terminals toch op de hoogte zijn van transmissies die gaande zijn) hebben we kunnen laten zien dat Frame Capture vooral belangrijk is bij het oplossen van collisions met hidden terminals. Collisions tussen coordinated nodes (nodes die dichter bij elkaar staan en elkaar wel kunnen horen) hebben meestal een signaalsterkte die min of meer gelijk is en kunnen daarom veel minder vaak door Frame Capture opgelost worden. Daarnaast hebben we kunnen laten zien dat de verhoogde coordinatie van nodes (de hidden terminals die stil zijn) er zelfs voor zorgt dat de performance van het netwerk iets afneemt omdat alle collisions tussen coordinated nodes zullen zijn, en deze niet door Frame Capture opgelost kunnen worden.

Acknowledgements

The road towards the work lying before you today has not always been paved with gold. A long time ago I started with the preliminary research for this thesis, while also working 2 days per week at a company and trying to do another Masters (the one in Computer Security). This proved to be very unfertile ground for decent and productive research, and it was only after I quit my job and stopped with my second Masters that my work on vehicular networks could really start. After that I fell into numerous other pitfalls that come with a research as big as this - I did not clearly define the scope of my work, leading me to investigate branch after branch of related topics - with the channel modeling and signal theory as the most time-consuming examples. However, after a year of hard work I'm proud of the result you have in your hands right now.

This work however would not have come to fruition if it were not for the people around me. I would like to thank my supervisors and professors Martijn, Geert and Georgios for their support all these months, their insights and guidance has always helped me steer in the right direction. In the early stages of my research Mark Bentum took the time to explain how to look at OFDM and what made these carriers exactly 'orthogonal', for which I owe him many thanks.

The people from the OMNeT++ Google Group have helped with the implementation issues I faced from time to time, thanks to a platform like Google Groups many people are helped forward with anything they might encounter.

Last but certainly not least, my gratitude goes out to my parents who always helped to motivate me and supported me financially during all these years that I received education, and my family and friends.

So long.. and thanks for all the fish!

Contents

List of Figures

List of Tables

1 Introduction

Nowadays, mobility and transport have become vital aspects of our society. Almost everybody has a car parked in front of their house or in their garage, which is used everyday - to get to work, to bring kids to school and pick them up, to visit family and friends, to buy groceries... these are just a few typical examples of how much we have gotten used and attached to our increased mobility and the convenience of owning a car. It is not even a luxury anymore; everybody has a car these days. With an increasing economy, both goods and people tend to move around a lot more and a lot faster than before, and the infrastructures we need to support that mobility are always expanding. But even though the infrastructure expands, it happens often that its capacity is not sufficient.

The traffic density on the Dutch highways is enormous, not just around the big cities, and on average working days there are over 200 km of traffic jams. This is about 4% of the entire Dutch highways network. Peaks in congestion due to weather conditions can rise up to 800 kilometers. The Dutch Ministry of Transport, Public Works and Water Management calculated that the annual economical damage because of traffic jams was between 2,8 and 3,6 billion euros in 2008. This was an increase of 7 percent compared to 2007, and even a 78 percent increase compared to 2000 [4]. These costs include both direct and indirect economical damage like extra fuel usage, the implied environmental footprint and commuters arriving late at work which causes companies to lose time and money. This does not only apply to the Dutch highways however, all growing societies around the globe suffer from heavy traffic congestion at peak hours.

Many solutions to this problem have been proposed. Usage of the roads is being taxed, people have been motivated to travel together instead of separately (also known as

carpooling) and in the Netherlands a system called *rekeningrijden* or 'billed driving' has been proposed to distribute the infrastructure maintenance costs more evenly over the people who actually use it. However, none of these solutions actually try to increase the capacity that the roads in terms of 'passing vehicles per hour' can sustain.

Vehicular networks try to fill that gap. By creating networks between vehicles, all users of the roads could have increased knowledge about the 'state' of the road, the current amount of traffic and the speed at which the traffic on it travels. This information can then be used to increase the safety on the roads and the capacity, leading to less traffic congestion, reduced emissions and reduced costs.

This research does not cover the entire challenge of how vehicular networks can make our lives a bit easier, but merely a few of the aspects of it; we look at some of the challenges presented when simulating vehicular networks. Many application protocols and systems that are to be used in vehicular networks must of course be tested first, at different stages during their development. Simulation of these applications and protocols is an important step in this process. We can learn a lot about the expected behavior and find out how things would function and whether they are scalable without implementing all functionality in real cars. Also, simulations reduce the need for testing the behavior on circuits or blocking stretches of the highway, which is of course an expensive undertaking.

1.1 Problem statement

This thesis thus focuses mainly on the simulation of the physical layer in vehicular networks. We give special attention to one problem in particular: Frame Capture. Frame Capture happens when two nodes try to send a frame to a third one at the same time and at the same frequency, i.e. when a collision occurs. Under some circumstances (depending on the signal to noise ratio's and the arrival times of both) the receiver is able to *capture* one of both frames, instead of losing them both because of the collision. This became the first important question that we tried to answer: How does Frame Capture work exactly? We looked at the knowledge that was already available and performed a literature study on three topics: basic IEEE 802.11 networks, the typical IEEE 802.11 physical layers and on Frame Capture. We looked specifically at the 802.11a and 802.11p physical layers because they are both based on Orthogonal Frequency Division Multiplexing (OFDM). As can be read later in Chapter 3.4, IEEE 802.11p is the variety that is used for vehicular networks, which makes it the most important physical layer for us. Based on that literature study, the following observations came to the surface:

- Frame Capture behaves differently on chipsets of different manufacturers [5, 6, 3].
- Because of its nature, Frame Capture probably occurs often in vehicular networks with a lot of broadcast traffic.

Much more information about IEEE 802.11 and Frame Capture will be presented later in this thesis. These two observations and the literature study also lead to my main problem statements:

- **Frame Capture is probably important for vehicular networks. However it is still unknown to which extent.**
- **IEEE 802.11p, the standard for vehicular networking, is poorly supported by almost all wireless network simulators.** There are quite a few network simulators around and some of them also support wireless communication protocols. However, the standard that will be used for vehicular networking, IEEE 802.11p [7], is relatively new and is not yet fully supported, not by any simulator currently available.
- **Frame Capture, although interesting, is not accounted for in the wireless network simulator that our research group normally uses (which is MiXiM [8], a wireless network simulator built within the OMNeT++ framework [9]).** Frame Capture is at best poorly implemented and in all situations poorly documented.

Also, when answering these questions regarding to Frame Capture and looking into our wireless network simulator, another problem arose:

- **MiXiM does not provide bit-error estimation functions in its physical layer implementation for 802.11a or 802.11p. The functionality present is only for 802.11b, which is fundamentally different [10].**

1.2 Research questions

Based on these problems, the following research questions were formulated:

1. **How can Frame Capture behavior in an IEEE 802.11p vehicular network be modeled?**
   How does Frame Capture behave in 802.11a networks? Will that be much different from 802.11p? It is necessary to account for the different behavior types found in different chipsets.
2. **How can the current IEEE 802.11 physical layer implementation in MiXiM be improved?**
   Which changes are needed to facilitate IEEE 802.11p communication and what important factors are not yet considered? In these changes to the physical layer, Frame Capture must also be considered.
3. **How does Frame Capture influence IEEE 802.11p network performance in a vehicular environment?**
   In what way does the road traffic density influence the network performance? Does

Frame Capture have a positive effect on the network throughput in a broadcasting environment (i.e. a network where most traffic is broadcast traffic)?

4. **What is the relationship between the hidden terminal problem and Frame Capture?**
   Does Frame Capture still occur if the hidden terminal problem were not present?

We try to answer the last two questions by performing simulations with the implementation resulting from the first two and in doing so, we tried to solve the problems mentioned and provide more insight in Frame Capture and its relationship to vehicular networks.

1.3 Outline

This thesis is structured as follows: in Chapter 2 we first provide basic background information about IEEE 802.11 wireless networks in general, look at the hidden terminal problem and at the reasons that collisions can occur in a wireless network. We then look at the physical layer by examining the inner workings of OFDM, the technology at the core of the physical layer of 802.11a and 802.11p. After this general introduction, Chapter 3 discusses vehicular networks. We look at what makes them different from the normal wireless networks, what their potential applications are and where the great challenges lie when dealing with vehicular networks. In Chapter 4 we discuss Frame Capture, what it is exactly, how and when it happens and we try to illustrate this with figures found in literature about the exact scenarios at which Frame Capture occurs. We also analyze the impact of Frame Capture on vehicular networks. Chapter 5 then provides more insight in our wireless network simulator and especially its physical layer - how it works and what we need to change to implement Frame Capture.

Chapter 6 shows how we implemented Frame Capture and what design choices were made. We validate the implementation in Chapter 6.3. The simulations we did to investigate the impact of Frame Capture in a vehicular network and the results of these simulations are described in Chapter 7. This thesis is concluded with a description of future work that can be done in this field and with the conclusions of my research.

2

IEEE 802.11 Wireless Networks

In this chapter we discuss various aspects of the 802.11 wireless networking family and the required specifics about the MAC and physical layers. We focus on the modulation and multiplexing techniques of 802.11a and 802.11p (OFDM). Readers already familiar with wireless networking can probably skip Sections 2.1 and 2.2 about the MAC layer, CSMA/CA and the hidden terminal problem.

2.1 Basics

IEEE 802.11 [11] is the family of wireless networking standards created and published by the Institute of Electrical and Electronics Engineers (IEEE). Figure 2.1 shows how 802.11 relates to the other IEEE 802 networking standards, it basically consists of a new Medium Access Control (MAC) layer and new physical layers, and offers an Ethernet-like network over a wireless medium to the higher layers. Note that 802.11 has quite a few different physical layers, they use different modulation techniques and various frequency bands. The MAC layer is roughly the same for all variations, however for 802.11p this is also changed to counter some of the issues mentioned in Chapter 3.3 related to authentication and association.

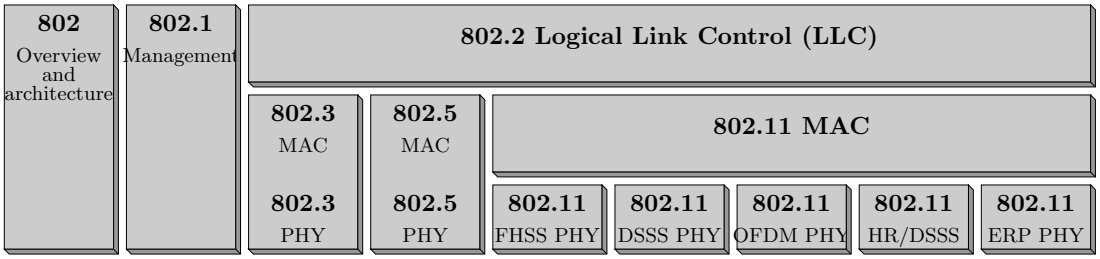


Figure 2.1: *The IEEE 802 family (Copied from [1], Figure 2-1).*

2.2 Medium Access Control

Because the wireless medium is a lot different from the wired medium used in for example 802.3 (Ethernet), a new MAC layer needs to be defined to successfully mitigate problems such as interference, collisions and the increased security vulnerability. Therefore, an adapted access scheme is used in this MAC layer: Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). This access scheme counters at least some of these problems. It contains a few different coordination functions, among them the Distributed Coordination Function (DCF). This is the most common one, and is the one relevant to discuss in this thesis.

The DCF organizes the access to the medium according to a set of basic rules.

- If the station wants to send a frame, it first senses the channel to see if somebody else is sending (this is called carrier sensing). If the medium is idle during a predefined period of time called the Distributed Inter-Frame Space (DIFS), the node can start sending right away. If the medium is busy, the station defers from sending and waits until the medium becomes available again. After the medium has become available, it again senses the medium for a DIFS.
- If during this second DIFS the medium remains idle, the station enters a so-called *contention window* or *backoff window*. This is a time window divided in slots. The window size (in number of slots) depends on previous transmissions, but has a minimum and maximum defined by the standard. In 802.11p the contention window size is between 15 and 1023 slots and always increments in powers of 2. Every slot has a fixed duration, it depends on the physical layer what the exact length is (in 802.11p it is 16 μs). The station randomly chooses a number of slots (within the contention window), that is the number of time slots it will wait before it actually starts trying to transmit. This means that if two stations were both waiting for a transmission to end, they will not start transmitting directly after the DIFS (and generate a collision). After picking a random number of (say n) slots the *backoff timer* starts counting down from n to zero, when it reaches zero the station can transmit.
- If, while waiting a random number of slots, the station senses another transmission,

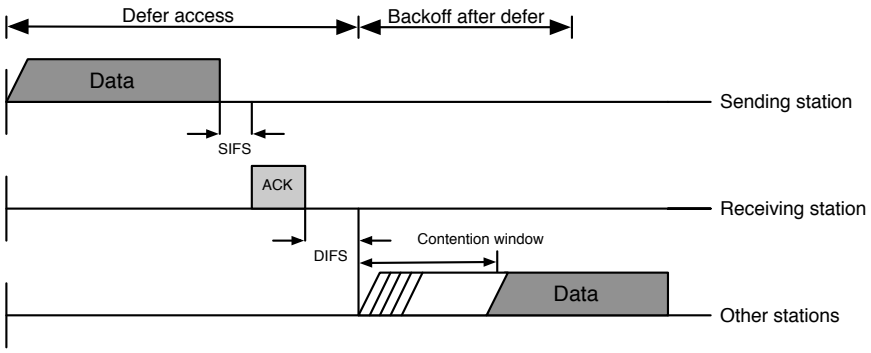


Figure 2.2: *The IEEE 802 DCF. (Based on [12], Figure 3-7)*

because another station was also waiting and picked a random number smaller than its own, it freezes the backoff timer. Then, after that transmission and a DIFS of idle time on the medium, it does not pick a random number again but merely restarts the backoff timer. This means that a station which had to wait the first round has a higher probability of gaining first access to the medium in the subsequent rounds.

- If the backoff timer reaches zero, the station transmits. If then the transmission fails (i.e. no Acknowledgement (ACK) is received), it doubles the contention window size and tries again. This means that when the number of nodes (and collisions) in the network increases, stations automatically start waiting longer and the algorithm remains stable.

- If a station needs to transmit a frame that, in the algorithm, logically follows the just transmitted frame (such as an ACK frame to indicate correct reception, or a new data frame if the entire frame is being fragmented), the station does not have to wait for a DIFS period. Instead, it can start transmitting after a Short Inter-Frame Space (SIFS), this guarantees that no other station that is not involved in the current transmission can seize the medium before it is completed. In this way multiple Inter-Frame Spaces are defined for frames with different priorities.

In Figure 2.2 part of the algorithm is shown in action. A sending station starts transmitting a data packet after waiting a DIFS, the receiving station replies after waiting a SIFS with an ACK, and after a DIFS and a random backoff period another station that also wanted to send a frame can start transmitting.

There are two ways in which a collision can occur with this DCF. If we assume that all nodes can hear each other, they will all be quiet during the transmission of another node. However if after a transmission two nodes choose the same (random) number of slots, they will start transmitting at exactly the same moment and their transmissions will collide. We call this a collision caused by a simultaneously ending backoff timer. The second way a collision can occur is because of the hidden-terminal problem, shown

in Figure 2.3. This happens when two nodes both want to transmit but cannot hear each other. This can happen for many different reasons; for example when there is an obstacle in between, or because they are simply too far apart. It does create possible collision scenarios though, because in network situations like the one in Figure 2.3 the carrier sensing mechanism alone is not enough. If either node A or C starts transmitting to B (because it thinks the channel is idle) while the other node is also transmitting, either to B or to another node in the network, their transmissions will overlap spatially at node B, causing B to perceive a collision of two signals.

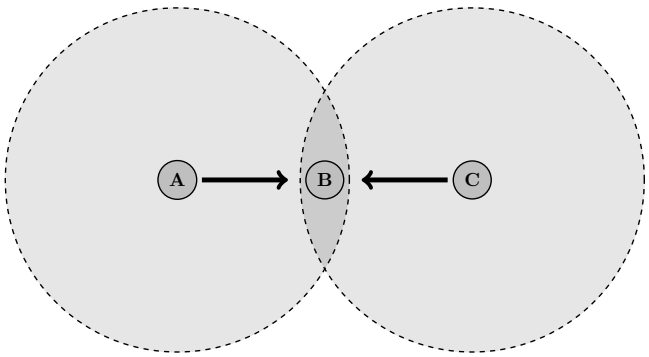


Figure 2.3: *Simplified visualization of the hidden terminal problem.*

To cope with this hidden terminal problem (HTP), the 802.11 DCF includes another feature: so-called Request To Send (RTS) and Clear To Send (CTS) frames. Using the standard waiting protocol described above, when a station obtains the medium it does not start sending its data but instead it first sends an RTS, a short frame asking for permission to send, containing basic information like sender, receiver and the time (duration) that the data frame will occupy the medium. The receiving station then has to reply with a Clear-To-Send frame, containing again the duration and the address of the node that is allowed to send [7]. In this case (looking at Figure 2.3 again), when A wants to send something to B and sends an RTS and B replies with a CTS, station C and any other station within transmission range of either A or B will be notified that the medium will be occupied for the duration defined in the RTS or CTS frame. Of course, A and C could still send an RTS at the same time that would collide at B, but then neither A nor C would receive a CTS and they would both know somebody else is trying to transmit, starting a random backoff procedure. This mechanism significantly decreases the collision-related throughput loss, especially for large data frames [1].

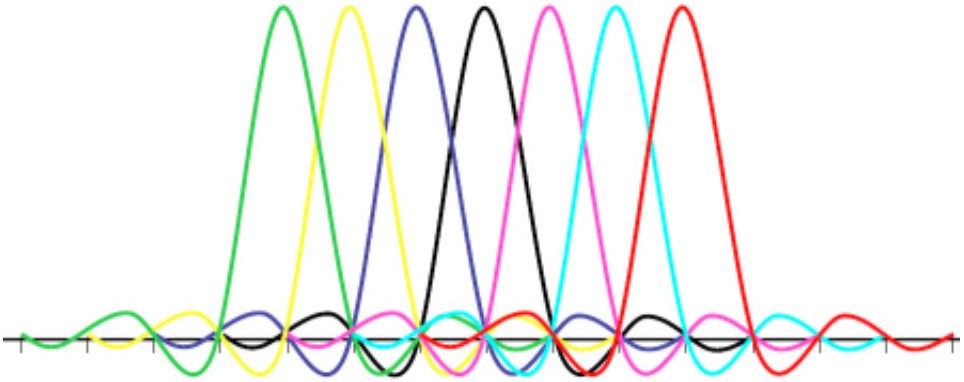


Figure 2.4: *Orthogonality of subcarriers in the frequency domain (Copied from [13]).*

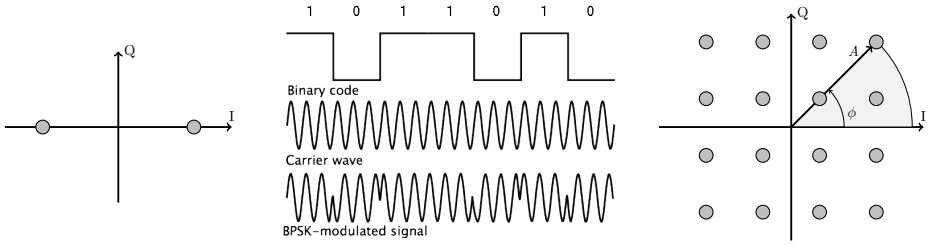
2.3 Orthogonal Frequency Division Multiplexing

2.3.1 Introduction

IEEE 802.11p uses a physical layer similar to 802.11a with OFDM as the primary multiplexing technique. In this section we will look into OFDM and describe how it works, so that the concept is clear in the later chapters about 802.11p and Frame Capture (Chapters 3.4 and 4).

Normal (single-carrier) modulation techniques use the whole channel and modulate data onto a signal at a high rate, with one symbol occupying the entire bandwidth for a very short time. OFDM however divides (or *multiplexes*) the channel in many small subcarriers and every subcarrier is modulated at a much lower rate. In order not to waste too much bandwidth on guard bands between these subcarriers, their frequencies are chosen in a such a way to create *orthogonality*; without spacing between the carriers they do not interfere with each other. This of course greatly improves the channel efficiency. In the power spectrum from Figure 2.4 we clearly see that at the peak of every subcarrier all other subcarriers are zero, i.e. at that frequency the other subcarriers do not contribute any energy to the signal. The orthogonality is crucial to OFDM, if subcarriers are not chosen orthogonal to each other, they will overlap and interfere significantly.

Apart from the multiplexing part of OFDM, the individual subcarriers are modulated to contain the data. Depending on the chosen modulation both amplitude and phase can vary in an OFDM subcarrier. The frequency however has to remain constant to preserve the orthogonality.



(a) BPSK constellation diagram (b) Example of BPSK modulation (Based on [14], Figure 9-15) (c) Explanation of the constellation

Figure 2.5: *BPSK and QAM constellations*

2.3.2 OFDM mathematics

In order to explain how these orthogonal subcarriers are exactly generated, we need to look a little bit into the mathematics behind OFDM. Let's first look at OFDM in the time domain:

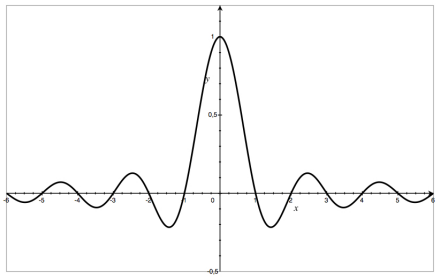
An OFDM subcarrier is a normal sine wave at a certain frequency, modulated with the data. As said before the frequency does not change, only the phase and amplitude can be modulated. Typical modulation schemes that achieve this are Binary Phase Shift Keying (BPSK), Quaternary Phase Shift Keying (QPSK) or Quadrature Amplitude Modulation (QAM). These modulations can be described in the complex plane as 'constellations' of points, where each point is the same sine wave but at a different phase and/or amplitude, and each point encodes a number of bits. This is illustrated in Figure 2.5. If the number of points in the constellation increases, more bits can be mapped onto each point. In Figure 2.5(a) for example, there are 2 points and only a 0 and a 1 can be mapped on both points. In Figure 2.5(c) each point in the constellation represents 4 bits, which means that there are $2^4 = 16$ points in total. In the complex field, a vector can be drawn from the origin to the point: the angle that the vector makes with the x-axis is the phase (ranging from 0 to 360°), and the length of the vector is the amplitude.

The desired orthogonality is created in two steps: carefully choosing the subcarrier frequencies and, after summing all the subcarriers, calculating the convolution (a type of integral transform) of the summed signal with a so-called rectangular window. A rectangular window is quite a simple signal: it is 1 between $-\frac{T}{2}$ and $\frac{T}{2}$, $\frac{1}{2}$ at the borders and 0 otherwise. The rectangular window used in OFDM is 1 between 0 and T (shown in Equation 2.1).

$$rect(t) = \begin{cases} 0 & \text{if } t < 0 \text{ or } t > T \\ 1 & \text{if } t > 0 \text{ and } t < T \\ \frac{1}{2} & \text{if } t = 0 \text{ or } t = T \end{cases} \tag{2.1}$$

This rectangular pulse in the time domain is quite obvious. When Fourier-transforming

$$\int_{-\infty}^{\infty} \mathrm{rect}(t) \cdot e^{-i2\pi ft}\, dt = \frac{sin(\pi f)}{\pi f} = \mathrm{sinc}(f)$$



(a) Fourier-transform of rectangular window (b) The resulting spectrum

Figure 2.6: *The Fourier transform of the rectangular pulse*

this rectangular pulse, we see that the spectrum of this rectangular is a *sinc* function (as in Figure 2.6(a)).

A *sinc* function looks exactly like the carriers in Figure 2.4, it has a peak at one specific frequency and is 0 at the center frequencies of the adjacent subcarriers, or in the time domain, at the nonzero integer values of t. This means that the rectangular pulse in the time domain looks like a *sinc* function in the frequency domain. Any signal that is convoluted with a rectangular pulse of the right width looks exactly like a subcarrier as in Figure 2.4. The width of the pulse defines the distance between the peaks in the resulting spectrum. This is valid with only one subcarrier, but by choosing the other subcarriers just the right way (explanation in Appendix B), summing them and after that calculating the convolution with the rectangular window creates an OFDM signal (convolution in the time domain is equal to multiplication in the frequency domain) where every subcarrier has one peak as in Figure 2.6(b), and is zero on the frequencies of all the adjacent peaks. This allows very closely placed subcarriers, increasing the spectral efficiency of the signal. The frequency difference between the subcarriers needs to be $f_c = \frac{1}{T}$. As an example, T in 802.11a is $3.2\mu s$, this corresponds to $1/3.2 \cdot 10^{-6} = 312.5$ kHz subcarrier spacing.

All the above would result in a spectrum (for 802.11) that looks like Figure 2.7. At the gaps in the spectrum are special pilot carriers, used for time and frequency synchronization purposes.

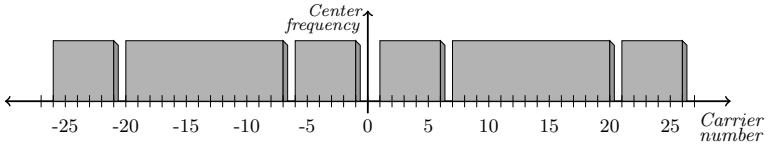


Figure 2.7: *802.11 OFDM channel structure (Copied from [1], Figure 13-8).*

So in short: all subcarriers of an OFDM signal simultaneously transfer a number of bits. The number of bits per symbol depends on the subcarrier modulation. The symbol rate

of the subcarriers themselves is low, but because OFDM uses many subcarriers the total data rate is equal to a single-carrier system.

2.3.3 Preamble detection

At the physical layer of 802.11a and 802.11p, Physical Layer Convergence Protocol (PLCP) handles the formatting of the frame. The PLCP preamble is an important phase of that frame and the reception process in general. A receiver goes through the following steps during the preamble:

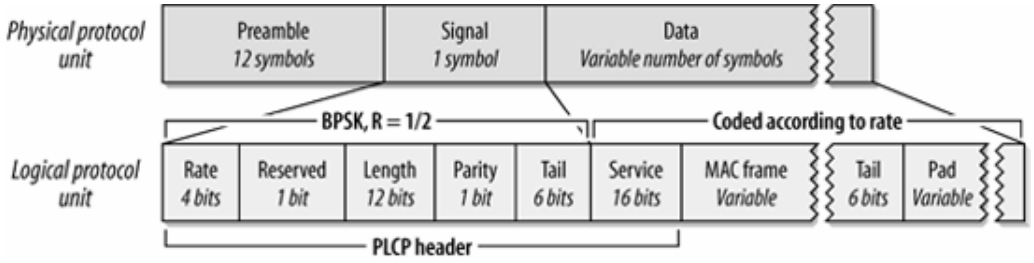
1. Detection and measurement of the signal power level; this has to be greater than or equal to RXSens, which is the receiver's minimum coding & demodulation sensitivity. Any signals with a power level below this threshold will be considered as noise. RXSens for Atheros chipsets is around -91 dBm [15].
2. Automatic Gain Control (AGC) during which it sets the signal gain to a level appropriate for the received signal power. This is necessary to have the signal arrive at the receiver circuitry with always the same power level.
3. Frequency synchronization using a Phase-Locked Loop (PLL) (see also Ch. 7 & 8 of [14]).
4. Timing synchronization [15].

A visualization of the PLCP frame and preamble is given in Figure 2.8. Note that the time values mentioned in Figure 2.8(b) apply to 802.11a. The preamble consists of 10 small predefined symbols followed by two large ones. The short ones are used for AGC and frequency synchronization and the two long ones are used for fine-tuning. In general the rule applies that the longer the preamble is, the better the receiver can estimate the channel and the better the receiver can lock to the signal. [1, 16].

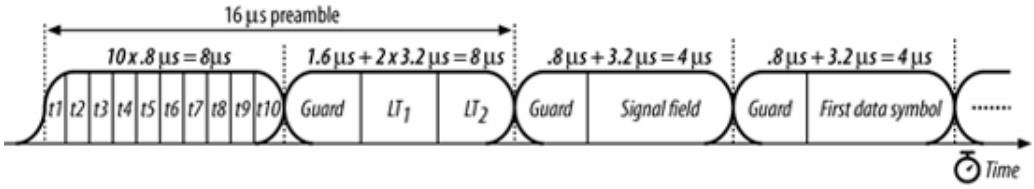
2.3.4 Guard times, coding and forward error correction

There are many types of interference and disturbances which can disrupt or destroy a signal in the wireless medium. Two of them are Inter-Symbol Interference (ISI) and Inter-Carrier Interference (ICI). ISI occurs when two subsequent symbols interfere with each other; this is caused by multipath propagation or frequency delays which disrupt the receiver's timing synchronization. ICI is caused when (perhaps in a narrow part of the frequency band) the carrier frequency shifts a little bit. This disrupts the orthogonality of the subcarriers and causes energy of one subcarrier to leak into another. This can be caused for example by Doppler shift, which can be a serious problem in OFDM.

To counter ISI, every OFDM symbol starts with a *guard time*; this is a small portion of the symbol time. The symbol time is divided in the guard time and the *Fast Fourier Transform (FFT) integration time* (also shown in Figure 2.9). This is important because



(a) PLCP Frame Layout



(b) PLCP Preamble (and first data symbols)

Figure 2.8: *PLCP Frame and Preamble for 802.11a (Copied from [1], Figure 13-14 and 13-15)*

multipath delay can cause various 'copies' of the original symbol to arrive at the receiver, also called different signal components. These components are generated by reflection of the signal against buildings, metal structures and other big objects. The Line Of Sight (LOS) component is the part of the signal that arrives directly at the receiver, and usually at various delays one or more multipath components show up, at a lower amplitude than the LOS component. If because of multipath delay a part of the previous symbol arrives during the guard time, this does not interfere with the symbol itself. This is only true however if the guard time is chosen such that it is bigger than the biggest reasonably expected multipath delay. More about multipath delay and multipath fading can be found in Appendix A and in Chapter 10 of [1].

The multipath components do not cause variations in the frequency, so the orthogonality is still preserved. However to preserve the orthogonality of the signal the receiver must have an integer number of cycles in the FFT integration time. This integer number is needed because the FFT works with discrete samples drawn from the signal, not the original signal itself. If in the FFT integration time there is not an entire cycle (360° in the complex field) or an integer multiple of that, the sampling stops at some signal value and the next sampled value (belonging to the next symbol) will differ greatly. This results in *discontinuities* in the signal, and this changes the perceived frequency of the subcarrier. This in turn destroys the orthogonality of the signal [1].

To prevent destroying the orthogonality, whatever is transmitted during the guard time should have the same frequency as the symbol itself. In that case when the signal shifts a bit over time, the FFT integration time still contains a nice (integer multiple of cycles) part of the signal. To achieve this the transmitter sends the last part of the symbol itself

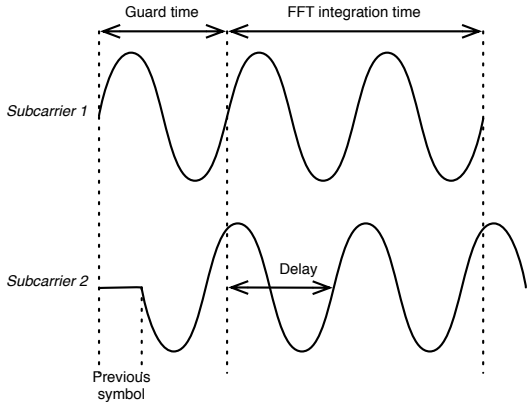


Figure 2.9: *Cyclic prefix extension (Based on [1], Figure 13-6)*

during the guard time (see Figure 2.9). This is called *cyclic prefix extension*. Also, in situations where there is no LOS component, the OFDM receiver could in the best case still figure out based on a multipath component what the original symbol was.

In order to further increase robustness of the signal, a typical OFDM transmitter also adds coding to the data. This is strictly not part of OFDM, but very commonly used in combination with it. The coding enables Forward Error Correction (FEC) at the receiver; with FEC a receiver can detect and correct errors without requiring retransmission. Coding works as follows: the encoder expands the original data bitstream, thereby adding redundancy. How much redundancy is added is determined by the *coding rate*, for 802.11 this varies between 1/2 and 3/4. A coding rate of 1/2 means that every original data bit is replaced with 2 coded bits. There are many different coding algorithms which vary in complexity, the one that is used in 802.11 is a so-called convolutional encoder. A convolutional encoder works on a continuous stream of bits; it takes m bits as input and produces n output bits, where the coding rate is m/n. It uses a transformation function that has a shift register with the last 7 input bits. On every step it computes n output bits based on the 7 bits in the register, and m bits in the register are replaced with new input bits. An example of a small convolutional encoder (with only 3 bits in the register) is given in Figure 2.10. In this encoder the values in the registers are summed to create the various output bits. The way the register bits are combined (or the *generator polynomials* for the various output bits) is critical to the error-correction properties of the encoder, they vary depending on the coding rate and the number of registers (or *constraint length*).

One can understand that using such an encoder, an input bit kind of 'spreads' itself over time, as it is still present in the shift register a few computation rounds later. Depending on the generator polynomials, a certain string of input bits of arbitrary length will always generate the same output bits. Therefore when decoding, depending on the incoming bitstream, a decoder can find out what the original data bits were. 802.11 recommends a decoder which uses the Viterbi-algorithm, which is a *maximum-likelihood* decoder. If

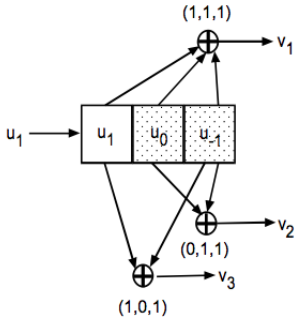


Figure 2.10: *An example of a convolutional encoder with 1 input bit, 3 output bits and 3 bits in the register. Please note that this is just an example of an encoder, the real encoder used for 802.11 is visualized in Figure 2.12.*

bit errors occur, the Viterbi algorithm outputs the most likely sequence of data bits, depending on the encoding and the occurrences of the bit errors. [1, 17]

One last smart step that the OFDM transmitter performs is interleaving: it spreads the coded bits over all subcarriers (1 bit at carrier 1, next bit at carrier 2, next at carrier 3 and so on), so that in the case of narrowband interference the bit errors are spread over the input bit stream. The probability of multiple bit errors close together (after de-interleaving) decreases, and if the decoded bits have errors, they can be more easily be corrected by Forward Error Correction because of the larger distance between errors. The receiver thus has a higher probability of finding the right data bits based on the encoded bits.

2.3.5 Strengths and weaknesses of OFDM

The multiplexing scheme used in OFDM has a few significant advantages over the single-carrier schemes used for instance in 802.11b. All the small subcarriers together generate a very flat, evenly distributed spectrum. The spectral efficiency is high; subcarriers are orthogonal to each other and placed very closely together. The various protection mechanisms make the signal very robust to narrowband interference and multipath fading, and ISI is effectively countered by the cyclic prefix extension (given that the delays are not much longer than the guard time). Also, the efficiency loss caused by this cyclic prefix is acceptable because of the low symbol rate. Another advantage is that OFDM can be implemented in transmitters and receivers using relatively simple components, no advanced equalization circuits are needed.

There are also disadvantages however; because of the dependence on orthogonality an OFDM channel is very sensitive to carrier- and Doppler shifts. These shifts can be caused by movement of the nodes and various channel effects (about which can be read in Appendix A). Also, summing of all the independent subcarriers may create a signal with some high (amplitude) peaks, which causes a high Peak-to-Average Power Ratio (PAPR).

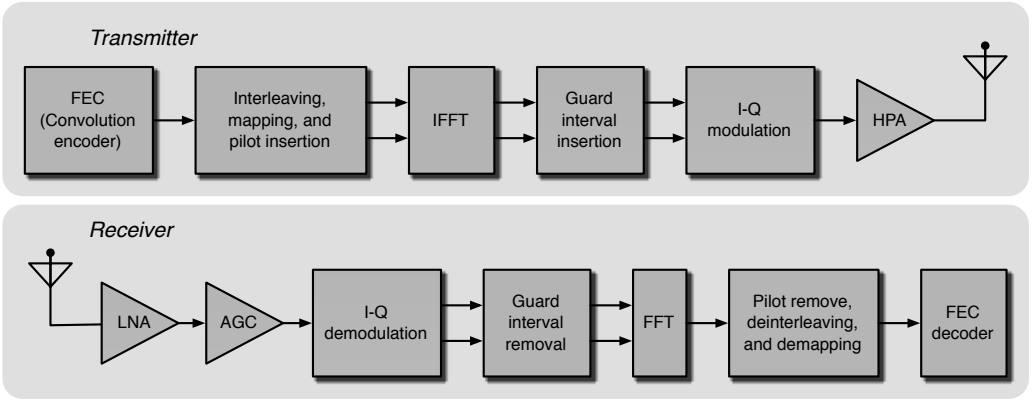


Figure 2.11: *An OFDM transceiver block diagram (Copied from [1], Figure 13-17)*

This increases the complexity at the transmitter, since practical Power Amplifiers (PA's) have a range at which they are linear (i.e. linearly amplifying the signal), and another part (near saturation) where they are non-linear. If the amplification is non-linear, this changes the form of the received signal at the receiver and thus increases bit errors [18, 19].

2.3.6 OFDM transmission and reception blocks

This section shows transmitter and receiver blocks for typical OFDM chipsets, and explains some of the steps that a transmitter goes through when transmitting data.

When looking at Figure 2.11, an OFDM transmitter does the following steps when transmitting (note that some of these steps are specific to 802.11) [1]:

1. Transmission rate selection. This depends on the channel conditions but is chipset-implementation dependent. The rate does dictate however which modulation is used and which coding rate. Together the modulation and coding rate determine how many bits are transmitted per symbol. Table 2.1 contains an overview of the used combinations.
2. Transmission of the PLCP preamble (at a fixed rate of 1 Mbps), these are a few long and short symbols to train the receiver and enable frequency synchronization.
3. Transmission of the PLCP header, also at 1 Mbps, which contains info about the frame that is about to be transmitted (frame length, encoding, transmission rate that will be used).
4. Creation of the data packet itself: some protocol-specific fields are added, the data is scrambled to prevent long sequences of zeros or ones and some padding bits are added. This data is encoded using the convolutional encoder described in Section 2.4.

5. Division of the coded bits into blocks (depending on the number of bits per OFDM symbol) to perform the interleaving process.
6. Pilot subcarrier insertion and using the Inverse Fast Fourier Transform (IFFT) generate the data signals.
7. Modulation of the subcarriers with the data signals using I-Q modulation. This modulation type is how QAM and QPSK signals are generated, for more information readers are referred to Chapters 9-5 and 9-6 of [14].
8. Amplification of the entire signal in a High-Power Amplifier (HPA) for transmission over the antenna.

When a frame is being received, the signal is first amplified using a Low-Noise Amplifier (LNA), and AGC is applied to get the signal to the (standard) power levels needed for demodulation. The rest of the reception process is basically the reverse of the sending process, as can be seen in Figure 2.11.

Table 2.1: *801.11a data rates and modulations / coding rates [1]*

Transmission rate (Mbps)	Modulation	Coding rate	Coded bits per carrier	Coded bits per symbol	Data bits per symbol
6	BPSK	1/2	1	48	24
9	BPSK	3/4	1	48	36
12	QPSK	1/2	2	96	48
18	QPSK	3/4	2	96	72
24	16-QAM	1/2	4	192	96
36	16-QAM	3/4	4	192	144
48	64-QAM	2/3	6	288	192
54	64-QAM	3/4	6	288	216

2.4 Bit Error Rate Calculations

This section discusses the calculation and prediction of bit errors of OFDM transmissions. OFDM uses BPSK and QAM [1, 14]; and as explained above, 802.11a and 802.11p are built on these modulation techniques. These bit error calculations do however apply to an Additive White Gaussian Noise (AWGN) channel. Bit error probabilities can be very precisely calculated for an AWGN channel, but we must note that the AWGN channel model is not ideal for a real vehicular network. Selection of the right channel model and performing the necessary calculations (e.g. incorporating the channel effects described in Appendix A) is a very challenging task that requires great knowledge of (antenna) physics and signal theory; therefore it is beyond the scope of this thesis. Using the AWGN channel model for bit error calculations gives us more insight into the matter however, and provides some reference for future work on this subject.

The aim is to use these bit error calculations in our implementation of 802.11p in the physical layer of our simulator. As known and described in Chapter 2.3, OFDM uses BPSK, QPSK and QAM as modulation techniques for various bit rates. The symbol error probability can be intuitively understood as the probability that noise or interference shifts the constellation point (see Chapter 2.3.2) so much that it ends up in another part of the constellation. The symbol error probability for BPSK [20] is shown in Equation 2.2.

$$P_s = Q\left(\sqrt{2\frac{E_{av}}{N_0}}\right) \tag{2.2}$$

In Equation 2.2, E_{av}/N_0 is the effective Signal to Noise Ratio (SNR) per bit. $Q(x)$ is the Q-function, which is the inverse of the normal cumulative distribution function of a Gaussian distribution (i.e. the integral that calculates the surface under the right tail).

$$Q(x) = \frac{1}{2}(1 - erf(\frac{x}{\sqrt{2}})) \tag{2.3}$$

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt \tag{2.4}$$

Since with BPSK there is only 1 bit per symbol, the symbol error probability is also the bit error probability. The symbol error probability for *M-ary QAM* in an AWGN channel is given in Equation 2.5.

$$P_M = 1 - (1 - P_{\sqrt{M}})^2 \tag{2.5}$$

$$P_{\sqrt{M}} = 2 \cdot \left(1 - \frac{1}{\sqrt{M}}\right) \cdot Q\left(\sqrt{\frac{3}{M-1}\frac{E_{av}}{N_0}}\right) \tag{2.6}$$

These formulas calculate the symbol error probability. The QAM constellation is Gray coded, which means that all adjacent QAM symbols are chosen such that they differ only in 1 bit [21]. Therefore if a transmitted symbol shifts to an adjacent one in the constellation, only 1 bit error occurs [20, 22], if the adjacent symbol represented a bit sequence that differed in more than 1 place, we would also have more than one bit error. That results in the following formula for the bit error probability:

$$P_b \approx \frac{1}{\log_2 M} \cdot P_M \tag{2.7}$$

If all bits were simply stuffed in OFDM symbols and afterwards extracted from them, we could calculate a packet error probability with these formulas. However, there are

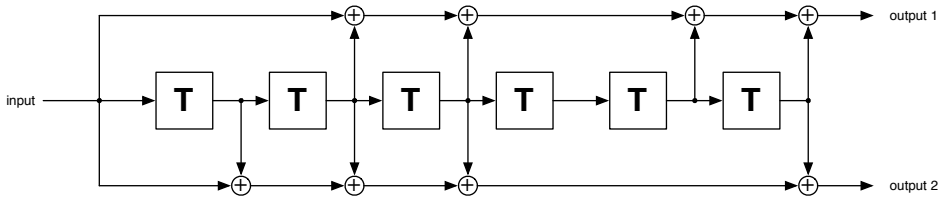


Figure 2.12: *The convolutional encoder used in 802.11a and p. It has generator polynomials* $g_0 = 133_8 = 91_{10} = 1011011_2$ *and* $g_1 = 171_8 = 121_{10} = 1111001_2$. *(Copied from [22], Figure 2.8)*

a few other elements to take into account. First: the E_{av}/N_0 value (the effective SNR) needs to be lowered, because not all the transmitted energy is stored in the information bits of the signal. A large cyclic prefix was added to counter ISI and ICI. The correct value for E_{av}/N_0 is given in Equations 2.8 and 2.9.

$$\frac{E_{av}}{N_0} = \alpha \cdot \frac{C}{\sum I + N} \tag{2.8}$$

$$\alpha = \frac{T_{IFFT}}{T_{IFFT} + T_g} = \frac{6.4\mu s}{6.4\mu s - 1.6\mu s} = 0.8 \tag{2.9}$$

Here T_{IFFT} is the Inverse Fast Fourier Transform period, equal to the symbol time, and T_g is the guard time, equal to the cyclic prefix length. C is the packet energy level and $\sum I + N$ is the sum of all interfering signals and noise. As we can see, all SNR values should be multiplied by 0.8 in and OFDM channel since 20% of the time there is no information transmitted [20].

Apart from this a normal 802.11a or p transceiver also performs coding and forward error correction on a signal. All input bits are encoded using a convolutional encoder with generator polynomials $g_0 = 133_8$ and $g_1 = 171_8$. This encoder is visualized in Figure 2.12.

Since the input bit stream is interleaved (consecutive bits are spread among all subcarriers), if a symbol error occurs the resulting bit errors are spread across the output bitstream. If they are separated enough, they can be corrected by the Viterbi decoder. We will not elaborate on all the details of error-correction here, they can be found in [23, 24].

Because of the error correction the packet error probability is not easily derived from the bit error probability. Instead, we can calculate an upper bound on the packet error probability, which is given by

$$P_e^m(L) \leq 1 - (1 - P_u^m)^{8L} \tag{2.10}$$

$$P_u^m = \sum_{d=d_{free}}^{\infty} \alpha_d \cdot P_d \tag{2.11}$$

$$P_d = \begin{cases} \sum\limits_{k=\frac{d+1}{2}}^{d} \binom{d}{k} p^k (1-p)^{d-k}, & d = \text{odd} \\ \sum\limits_{k=\frac{d}{2}+1}^{d} \binom{d}{k} p^k (1-p)^{d-k} + \frac{1}{2}\binom{d}{d/2} p^{d/2}(1-p)^{d/2}, & d = \text{even} \end{cases} \tag{2.12}$$

Here d_{free} is the minimal free distance of the convolutional code (10 in this case), α_d is the total number of errors with weight d and P_d is the probability that such an error occurs [20, 25]. This is also further explained in Chapter 8-2 of [24]. All these formulas combined yield the numerical results shown in Figure 2.13.

From all the equations and formulas shown above, we can conclude that there are many factors involved when determining the success probability of a transmission. We tried to summarize this here:

- Modulation plays an important role. If there are many bits in one symbol, the amount of noise or interference that a receiver can tolerate becomes less.
- Coding has a positive effect on the success probability in general. By adding a few redundant bits to the stream, bit errors can be corrected up to a certain point. This is complex mathematics though, we haven't investigated all the details of error-correction coding.
- The transmission rate is derived directly from the modulation and the coding rate. If a higher-order modulation is used that puts more bits in one symbol, the transmission rate increases. The coding rate defines how much redundancy is added and decreases the transmission rate, e.g. if a coding rate of 2/3 is used, one in every 3 bits is redundant, reducing the number of 'real' information bits.
- As can be seen in Figure 2.13, the length of the frame also influences the success probability. If we assume an equal error probability for every (decoded) bit, the chance of having one error in a frame increases with packet length. However the channel has a so-called *channel coherence time*, which is the time that the channel is assumed constant by the receiver. Depending on the channel, a frame cannot be longer than a certain length, otherwise bit errors do start to increase. More about this can be read in Section 3.4.2 and [20].

The results from Figure 2.13 are calculated for 802.11p, but in the equations above the frequency of the signal is not a variable, and neither is the symbol time. In Equation 2.5 it can be seen that the uncoded bit error probability is based only on the SNR per bit ($E_{av}/N0$). If the same coding is applied, the coded bit error probability and packet error probability are the same for an AWGN channel, regardless of the frequency of the

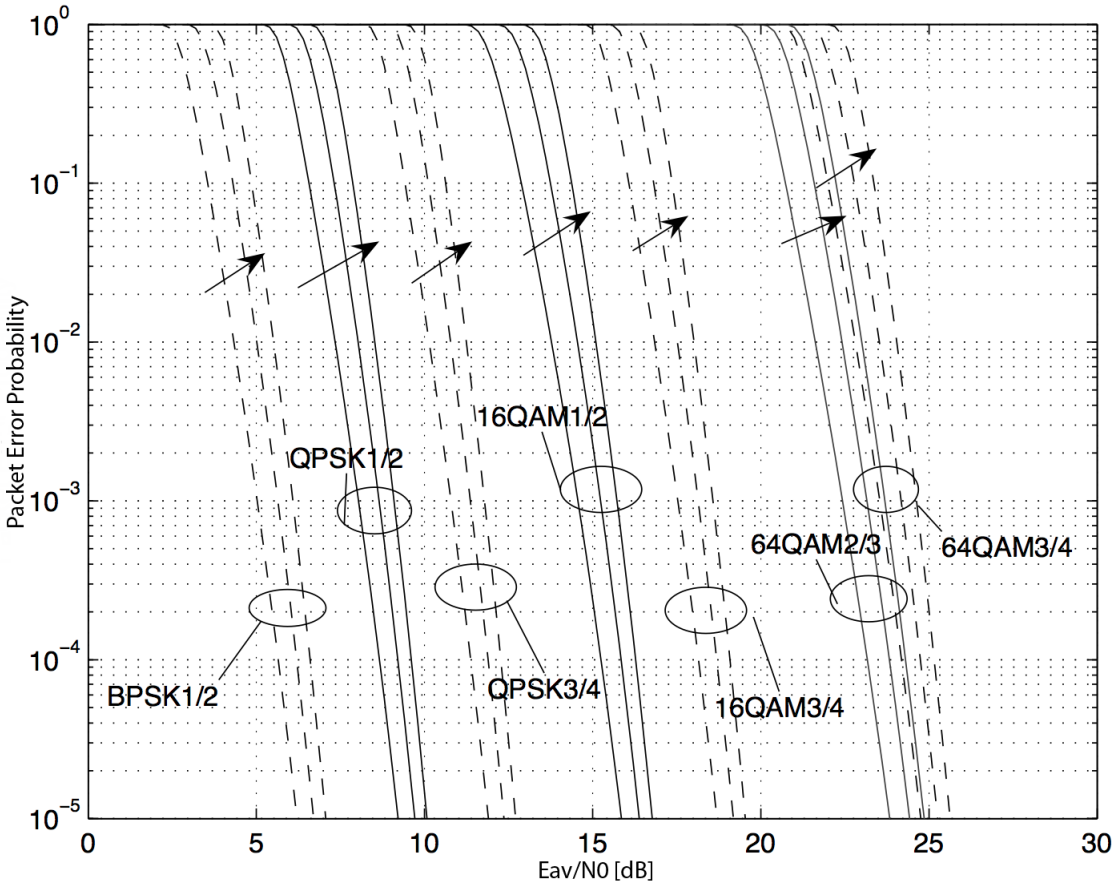


Figure 2.13: *Numerical results for the packet error probabilities of all QAM variations for packet sizes of 39, 275 and 2304 bytes. The arrow indicates the increasing packet size. Note that these results apply to an AWGN channel. (Copied from [20], Figure 5).*

channel. This means that these formulas apply to any OFDM-based system with the same modulations (BPSK, QPSK, QAM) and coding as used in 802.11a and 802.11p. It must be noted however that the frequency does have an impact on the observed signal-to-noise ratios in a channel; since the absorption of the channel increases with higher frequencies (see also Chapter 14-7-2 and Figure 14-4 of [14]).

3

Vehicular Networks

This chapter discusses some of the basics about vehicular networks; why do we want them, what can they do for us and what do we need to take into consideration when designing a vehicular network?

3.1 Basics

Vehicular networks are networks between vehicles, other vehicles and in some cases roadside infrastructures. A vehicular network can enable a wide variety of applications, both user-oriented and safety- and vehicle-oriented. There is a lot of similar terminology describing the same concepts, a the most commonly found terms are [26]:

- Intelligent Transport Systems (ITS)
- Inter-Vehicle Communication (IVC)
- Vehicular Ad-Hoc Networks (VANETs)
- Vehicle to Vehicle (V2V) Communication
- Vehicle to Infrastructure (V2I) Communication

This terminology also clarifies some of the different approaches that can be taken; a vehicular network can be set up as purely ad-hoc, or in some ways relying on supporting road-side infrastructure. In an ad-hoc network vehicles do not need an external infrastructure, they simply exchange information when they are closely together. In a V2I infrastructure the roadside devices can take up tasks such as traffic aggregation

and processing, access to larger (backbone) networks such as the internet and security features such as encryption key distribution. Both systems have many advantages: an ad-hoc network is easy to set up, can be deployed anywhere and is only present when there is traffic and is, up to a certain degree, very scalable, i.e. it does not depend on the capacity of the roadside infrastructure. There is a trade-off however, routing in an ad-hoc network is very hard and reliability depends greatly on the number of vehicles within range. If there are too few cars information might get lost, and if there are too many cars sophisticated algorithms are necessary to prevent the network from saturating. Also, some applications such as regular internet connectivity are not possible with a pure ad-hoc network since the vehicles only communicate with each other. A V2I infrastructure is more costly to deploy, but mitigates many of the disadvantages found in ad-hoc networks.

3.2 Applications

A few examples of user-oriented applications are normal internet connectivity for drivers, instant messaging, automatically updating the maps of satellite navigation devices, automated route planning and any internet-based personal entertainment as can be found in modern smartphones and tablets. However, the safety- and vehicle-oriented applications of vehicular networks are a lot more interesting from an academic perspective. There are many examples:

- **Cooperative Adaptive Cruise Control.** Some car brands already have Adaptive Cruise Control (ACC), a safety feature where the car measures the distance to the vehicle in front of it, and if that becomes too small (or decreases too fast), the car automatically starts braking, whether the driver is pushing the brakes or not. Cooperative Adaptive Cruise Control (CACC) achieves the same by letting the cars exchange messages when they brake, controlling and coordinating the longitudinal movement (and acceleration and deceleration) of an entire stream of vehicles. It has been proven that the response of a CACC system is much faster than its non-communicative ACC counterpart [27]. Note that CACC is not a pure vehicle safety application - it is also about comfort for the driver and about improving the efficiency of the traffic on the road.

- **Assisted merging.** On a normal highway a badly performed merge can have a significant negative impact on the flow of traffic on the highway. A driver assistance system could be developed where a vehicle merges automatically by exchanging messages with the cars already on the highway, creating a seamless merge which guarantees a smooth flow of traffic. Traffic theory research indicates that minor changes in speed on a saturated (road) network can lead to major traffic disturbances a little while later [28].

- **Traffic flow optimization.** If vehicles on the highway would be able to coordinate their speed with all other vehicles around them using V2V communication and make intelligent decisions regarding their speed, not just based on the 10 vehicles in front of the driver (what the driver can see), but based on all vehicles in the coming few kilometers, the flow of traffic would be a lot smoother, resulting in increased capacity of the road network, fewer traffic jams and major decrease of economical damage because of traffic congestion.

Our research group focuses mostly on the safety- and efficiency applications of vehicular networks. It can be easily understood that for safety-critical applications direct communication between vehicles provides many advantages over relaying traffic via roadside units, so for these kind of applications at least some form of ad-hoc networking must be present. This thesis therefore also focuses on the ad-hoc nature of vehicular networks. Ad-hoc network connectivity does not exclude V2I communication however, but we focus on the former. The standard (to be discussed in Chapter 3.4) however accounts for both.

3.3 Challenges

In a vehicular networking environment, two typical scenarios need to be considered primarily: urban and highway environments. In both cases the 'ad-hoc' characteristic of the network is quite extreme. On a highway vehicles don't always stay close to each other for a long time, and all vehicles travel in the same direction (if assumed that the vehicles traveling in the opposite direction form a separate network), but at every fork or exit nodes can enter or leave the network. It is thus incorrect to assume that a node in the network is still there if it was 60 seconds ago. This is even more true in an urban environment, where the vehicles have an even higher degree of freedom of movement [29, 30].

In such an environment, it is absolutely impossible to use a standard 802.11a, b or g wireless (ad-hoc) network. Tasks like authentication, updating routing tables and sending packets to other nodes are very unreliable because the network topology changes too much in very short periods of time. Typical Access Point (AP) scanning costs between 70 and 600 ms, authentication between a few ms up to more than a second, depending on the type of authentication, and association costs around 15 ms but is quite vendor specific. Also the association time needed increases if the user's sessions need to be handed over from another AP to the next one [31]. If for example a vehicle moving at a typical highway speed of 100 km/h passes a roadside access point, it is hardly authenticated to that AP the moment it is already leaving the AP's range. Relative speed differences with cars moving in the same direction will be smaller, but the environment is still challenging; more about this can be read in Chapter 5.2.1, Appendix A and in [2, 26].

Apart from the limitations from the MAC layer such as authentication, the physical layers from a standard Wireless Local Area Network (WLAN) are also not designed to handle that much movement. One of the normal WLAN variations, 802.11a, uses OFDM which is very sensitive to Doppler-shift (see also Chapter 2.3.5). If no effective countermeasures are taken, Doppler shift will significantly influence the efficiency and throughput of the vehicular network [32, 33]. Altogether many factors need to be considered in a vehicular network that are not that relevant in normal 802.11 infrastructure or ad-hoc networks.

3.4 IEEE 802.11p

IEEE 802.11p is the amendment to the 802.11 wireless networking family that will facilitate vehicular communication and provide solutions to many of the problems mentioned above. It specifies many of the needed adaptations, one of the main concepts they introduced is called Wireless Access in Vehicular Environments (WAVE). The standard has only recently been published (July 2010) and has been under development since 2006. The amendment only specifies changes at the MAC- and physical layers, higher layer protocols such as TCP and IP are considered out of the scope of the 802.11 standard. Considering the MAC layer, a Station (STA) can operate in WAVE mode, and when it does it can send messages to any other STA in WAVE mode with a valid MAC address, including group addresses, without first joining a Basic Service Set (BSS). If communication with a Distribution System (DS), a fixed road-side network is necessary, stations can also join a WAVE Basic Service Set (WBSS). Authentication or association is not required in WAVE mode, only the first step (joining the BSS) has to be performed. After that, data is sent directly to the DS [7].

3.4.1 Physical layer

The changes to the physical layer in 802.11p are a lot more interesting to us. The frequency band at which 802.11p will operate is in the 5.9 GHz ITS channel in the US, this band has been designated by the Federal Communications Commission (FCC) for use by Intelligent Transport Systems. The European telecommunications authority, or Conférence Européenne des administrations des Postes et des Télécommunications (CEPT), has designated a similar frequency spectrum after extensive studies [34, 35].

To counter the increased vulnerability of 802.11p (also see Chapter 2.3.5), such as the increased amount of Doppler shift due to movement, some physical layer parameters are changed in the 802.11p standard. One 802.11a OFDM channel uses 52 subcarriers out of which 48 are used to transmit data and 4 are pilot carriers. 802.11p uses the same number of subcarriers, but uses a smaller bandwidth per channel; 10 MHz opposed to 20 MHz in 802.11a. This also means that all parameters in the time domain (guard time, symbol time) are doubled compared to 802.11a. This has as a result that Doppler

spread decreases (due to the smaller frequency bandwidth), inter-symbol interference is also decreased due to the longer guard times. These doubled parameters in the time domain halve the effective data rate (3 to 27 Mbps against 6 to 54 in 802.11a) [36]. The standard also specifies more stringent Adjacent Channel Rejection (ACR) requirements, because the guard bands between channels are smaller. Table 3.1 contains a comparison of various 802.11a and p Physical Layer (PHY) parameters.

Table 3.1: *802.11a and 802.11p PHY values [2]*

Parameter	**802.11p**	**802.11a**
Channel bandwidth	10 MHz	20 MHz
Data rates	3 to 27 Mbps	6 to 54 Mbps
Slot time	16 μs	9 μs
SIFS time	32 μs	16 μs
Preamble length	32 μs	20 μs
PLCP header length	8 μs	4 μs
Air propagation time	< 4 μs	$<< 1$ μs
CW_{min}	15	15
CW_{max}	1023	1023

3.4.2 Effects of changes at the physical layer

The changes made to the physical layer have been chosen such that communication is still possible using OFDM, in spite of the increased amount of movement, Doppler shift and the challenging channel characteristics. The typical rms delay spread for a non-LOS signal component can be up to 400ns [37], this can more or less be interpreted as (quadratic mean of) the time difference between the LOS component and the last non-LOS component. The guard time is 1.6 μs, so the guard time is enough to eliminate ISI. However, according to [20], some efforts are still needed to protect the physical layer against the short channel coherence time and the increased amount of mobility. If we assume a maximum speed of 500 km/h, the maximum Doppler spread will be 2.7 KHz and that indicates a channel coherence time of 157 μs. A frame that is transmitted must not be longer 'on the air' than this period, or the training symbols at the start of a frame will not be long enough and the signal can become too distorted. The frame size that this relates to depends on the bitrate, but this is only 471 bits for the lowest data rate ($3\text{Mbps} \cdot 157\mu\text{s}$) [38].

Since the bit error probability (in an AWGN channel) of a communication system relies only on the SNR per bit or E_b/N_0 (Chapter 5-2-1 of [24]) and not on any other parameters, the only main difference that we need to consider is the doubled energy per bit in 802.11p. The symbol time in 802.11p is doubled and the data rate is halved (see also Chapter 2.3). If the data rate is halved and the transmit power remains the same, one symbol contains twice the energy it does in 802.11a, meaning doubled energy per

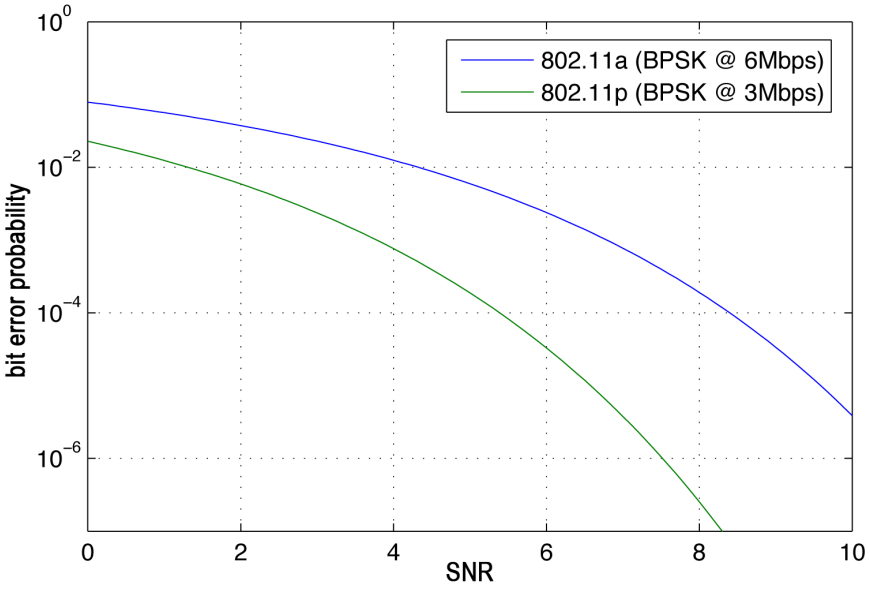


Figure 3.1: *BER curves for uncoded BPSK with 802.11a and 802.11p.*

symbol and doubled energy per bit (Equation 2.2. The resulting difference in (uncoded) bit error probabilities is shown in Figure 3.1, we can see that a doubled E_b/N_0 value implies a significantly lower bit error rate. The difference (in energy per bit) between both physical layers that is needed to achieve the same uncoded Bit Error Rate (BER) is exactly 3 dB.

The error-correction coding is exactly the same in 802.11p as in 802.11a, it introduces another change in the energy per bit and in the final bit error probability (Chapter 1.9 of [39]). While coded and uncoded channels are hard to compare, using a convolutional or block code results in a coding gain of a few dB that is independent of the SNR or E_b/N_0 [39, 23]. We thus expect the resulting coded BER curve to show a similar shift to the left as the uncoded BER curves in Figure 3.1.

4

Frame Capture

Now that we have covered all the basics about 802.11 communication and the challenges with vehicular networks, we focus on the phenomenon that we try to research with this thesis: Frame Capture. This Chapter covers all aspects of it; what it is, when and how it occurs and what its potential impact is on vehicular networks.

4.1 What is Frame Capture

Frame Capture occurs under some conditions when two transmissions overlap both spatially and temporally. When a receiver detects two frames at the same time this is generally regarded as a collision - if two sources send an electromagnetic signal at the same frequency they will interfere with each other and the receiver will not be able to decode either one of them. This is absolutely true, however, the last years it appeared that this is not totally black-and-white: under some circumstances one of the frames that is being received can still be decoded correctly even if another transmission is occurring simultaneously [5, 40]. This phenomenon of being able to receive a frame even in the presence of another frame is called Frame Capture (FC), sometimes also Physical Layer Capture (PLC).

4.1.1 Related work

Frame Capture has been observed and investigated in many different papers. [41] discusses the implied unfairness; that closer nodes can effectively 'capture' the channel if

their signal is sufficiently stronger and the consequences of that for ad-hoc networks. . [42] tries to further model that behavior and mentions a US Patent for a system that explicitly defines capture behavior [43]. [44] also discusses that patent and proposes the reordering of messages to enable better spatial concurrency and allowing more frames to be sent simultaneously through the medium. The authors of [6] refer to Frame Capture as *Physical Layer Capture*, and point out that the effect is larger at lower bitrates. They perform simulations in QualNet to support that. The authors from [3] perform the most valuable real-world experiments, investigating the behavior in great detail, and adding more details to the notion that apart from the signal-to-noise ratio the arrival time also plays a role in whether a frame can be captured or not. The work in [15] describes newer results and is from the same authors as [3], however they perform experiments with wired topologies. How they simulated the wireless medium with wires is not entirely clear. Also, most of these experiments and observations were done with 802.11a networks. Most of the insights and descriptions in this Chapter come from [3].

4.1.2 When does it occur

Normally the 802.11 coordination function (CSMA/CA, also see Section 2.2) prevents most collisions by using RTS/CTS and the random backoff counter. This only works with unicast traffic however; if a node wants to send broadcast traffic the RTS/CTS mechanism cannot be used, obviously because one node needs to respond to the RTS message. If multiple nodes respond to one RTS message a collision will already occur even before the actual data is being sent. The absence of RTS/CTS with broadcast traffic means that nodes are less aware of the traffic surrounding them and ongoing transmissions are more likely to be disturbed by hidden terminals.

4.1.3 How does it work

There are quite a few papers on the observed behavior that stems from Frame Capture, but unfortunately there is not a single one that explains at circuit/hardware level how the receiver deals with two (or more) interfering signals. Also, the relevant manufacturers (Prism and Atheros [45, 46], as also mentioned below) do not provide data sheets explaining their hardware with the level of detail that we require. It is therefore feasible to describe Frame Capture at a behavioral level, but explanations of the inner workings of the receiver circuitry are very hard to find and have thus been left out of the scope of this thesis. From a behavioral perspective, we can easily observe that there are certain thresholds involved, which we will explain further in this section.

Frame Capture can occur in many different situations, and there are many variables which influence the behavior and possible outcome of a collision. If two frames are being transmitted simultaneously, the following factors are important:

- *Exact time of arrival*
  The exact time difference between the arrivals of two colliding frames is an important factor. The timing difference we talk about here is in the order of microseconds. If the medium is busy the CSMA/CA mechanism implies a slot synchronization which will cause the frames to arrive more or less at the same moment. This can happen because the backoff counters of both sending nodes ended simultaneously (see also Chapter 2.2). If the interference comes from a hidden terminal a colliding frame could arrive at any moment during reception, because a hidden node that does not sense ongoing transmissions (and hasn't sensed any transmissions recently either) might or might not be synchronized to the time slots of the other nodes, and could be allowed to start sending at any moment.

- *Signal strength*
  Apart from the timing, one of the two signals has to be sufficiently stronger than the other for the receiver to be able to decode it even in presence of the interference of the weaker signal. If both signals are equal in Received Signal Strength (RSS), it really depends on the chipset whether it is able to decode either one of them. Also, the required difference in RSS depends on the timing relation. If the stronger frame arrives before the weaker frame, the stronger frame will probably be received normally, and the required SIR difference is quite small. If one of the two frames interferes with the other one's preamble (which is an important phase of the reception process: during the preamble the receiver tries to lock onto the signal), the required difference in RSS is a lot higher because it is harder for the receiver to correctly lock onto one of the signals.

- *Chipset manufacturer*
  It appears that even chipsets from different manufacturers behave differently under the above mentioned timing relation. In a Prism chipset [45] for example, the effect only occurs if the second, stronger frame arrives before or during the first frame's preamble. However in Atheros chipsets [46] capturing a stronger frame will occur also after the reception of the weaker frame's preamble (i.e. almost independent of the timing relation between the two arriving frames [47, 40]).

4.1.4 Frame Capture in other communication systems

Frame Capture in other communication systems has not gotten much research attention. It is known that analog systems can be designed in such a way that a receiver can tolerate a great deal of interference, for example traditional FM radio receivers can block out an entire interfering signal as long as the desired signal is only a little bit stronger. This is thoroughly described in Chapter 8-6 of [14], where the reported increase in SNR can be as much as 20 dB. They call this effect *FM quieting* or the *FM capture effect*, because of the analog nature of the system 'Frame Capture' is not a correct description of the phenomenon anymore. This is essentially what also happens in 802.11 networks though; the interfering signal (or noise) is suppressed in favor of the desired signal. [14] also

speaks of an SNR threshold that must be passed before an FM signal can effectively be captured. This threshold is around 10 dB. It is safe to assume that the same noise suppression circuits can also be used in digital FM (frequency modulation) systems, however clear literature on this topic is scarce.

4.2 Capture Scenarios

Based on literature [42, 3, 5, 15], we have distinguished the following relevant scenarios where two frames arrive almost simultaneously at the receiver. In this classification overview we assume that we have two frames (F_1 and F_2) and that F_1 is always the strongest. All scenarios where F_2 would be stronger always have their equivalents in Table 4.1.

In advance it is impossible to say which of the two frames is the desired one, so we just assume that the stronger frame is the one we want to receive. We cannot make this distinction based on the content of the frames, but receiving the stronger frame is also desirable because a stronger SNR means a lower BER, which increases correct packet reception probability. In a vehicular networking context, receiving the strongest frame is strongly related to receiving the frame of the nearest vehicle. This is also a desirable feature, because the nearest neighboring vehicles pose the greatest 'threat' to the vehicle itself. Just to provide an example: safety messages like emergency braking and other problems should really be received first. It is thus quite reasonable to assume that we want to receive the strongest frame.

Table 4.1 illustrates the Frame Capture scenarios with different timing and SNR situations. Δt is the exact difference between the arrival times (in μs), $L_{preamble}$ is the length of the frame's preamble (32μs in 802.11p), and the SNR (Signal to Noise Ratio) is $RSS_{F1} - RSS_{F2}$. As said, in all cases $RSS_{F1} > RSS_{F2}$. The exact layout of a frame is shown in Figure 2.8. A receiver is 'locked' to a frame if it has received the preamble of that frame and it is currently demodulating and receiving that frame, thereby ignoring the interference and regarding any other frames currently on the medium as noise.

Basically Table 4.1 tells us the following: a stronger frame can always be captured (and thus received correctly if no other interferers arrive) in the presence of another frame, if the SNR of the stronger frame is high enough. The required SNR is higher if Δt is less than $L_{preamble}$, because the frame's preamble is an important phase of the carrier detection and lock-on process (see also Chapter 2.3.3). If the receiver is locked on to the weaker frame, it is a bit easier to receive a stronger frame if it arrives, because the ability to receive a signal means that the receiver is better able to suppress it as well [44].

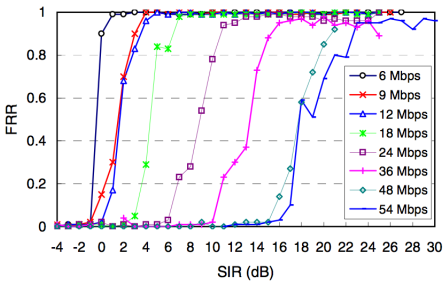
Figure 4.1 from [3] shows the required SNR in more detail. It shows the Frame Reception Ratio (FRR) on the y-axis for various SNR values and for various situations. In these experiments a testbed was created with a receiver, two senders (one normal sender and

Table 4.1: *Frame Capture scenarios (data rate of 6 Mbps). These timing relations and results are for Atheros chipsets [3]. The experiments were performed with 802.11a.*

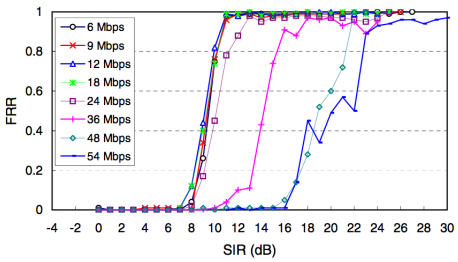
	Timing relation		Result
1.	$\Delta t > L_{preamble}$	P Frame 1 P Frame 2	Frame 1 is captured if $SIR > \sim 0\ dB$
2.	$\Delta t < L_{preamble}$	P Frame 1 P Frame 2	Frame 1 is captured if $SIR > \sim 12\ dB$
3.	$\Delta t < L_{preamble}$	P Frame 1 P Frame 2	Frame 1 is captured if $SIR > \sim 12\ dB$
4.	$\Delta t > L_{preamble}$, receiver locked on to Frame 2	P Frame 1 P Frame 2	Frame 1 is captured if $SIR > \sim 10\ dB$
5.	$\Delta t > L_{preamble}$, receiver NOT locked on to Frame 2	P Frame 1 P Frame 2	Frame 1 might be captured, but SIR should be at least ∼20 dB. Probability increases linearly as SIR increases.

an interferer) who cannot hear each other, and two extra receivers to monitor the two senders and get exact timings on the sent frames (see also Figure 1 in [3]). When experimenting with the induced collisions at the receiver, the authors make three distinctions; Sender First capture (SF) capture, Sender Last, Interferer Clear (SLC) capture and Sender Last, Interferer Garbled (SLG) capture. The Sender in this case is the sender of the desired frame, the other sender is designated as the interferer. In the SF case the strongest frame arrives first (corresponding to Scenario 1 and 2 in Table 4.1), in the SLC case the strongest frame arrives second while the receiver is decoding the weaker frame (Scenario 3 and 4), and in the SLG case the strongest frame arrives second while the receiver is, for any reason, not decoding the weaker frame (Scenario 5). The probability of success when capturing depends a lot on the data rate and the signal strength of the stronger frame, but at 6 Mbps in the SF case almost all frames can be captured even if the SNR is around 0 dB (both frames equal in signal strength). For higher data rates the required SNR also increases. It is clear to see that in the SLC case (Figure 4.1(b)) the required SIR is a lot higher, but does not change much if the data rate increases - only at the highest data rates does the required SNR also increase. If the receiver is not locked on to the weaker frame (or 'interference'), the data rate becomes even less important - in the presence of the interfering signal energy, the desired signal must simply be strong

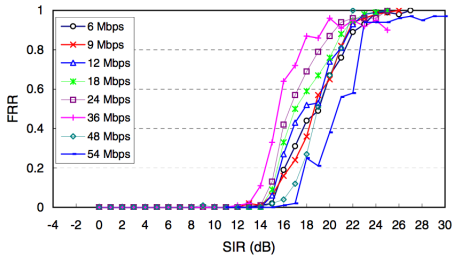
enough to be received. For a Frame Reception Ratio of 80 % the Signal to Interference and Noise Ratio (SNIR) needs to be around 20-22 dB; it depends on the requirements of the application whether that is sufficient.



(a) SF (Sender First) case



(b) SLC (Sender Last, Clear interference) case



(c) SLG (Sender Last, Garbled interference) case

Figure 4.1: *Required SNR for various Frame Capture scenarios. (Copied from [3], Figure 7)*

The main reason that in Figure 4.1(b) there is such a large initial threshold that must be passed before a frame can be received correctly is the correct reception of the preamble (see also Chapter 2.3.3). During the preamble a receiver cannot tolerate a lot of interference because the preamble dictates for a very large part how well the receiver is synchronized to the sent frame. At all the lower bitrates the required SNR for correct preamble detection is higher than the required SNR for the rest of the frame. From Figure 4.1(c) we can see that if a receiver was not able to lock onto a frame (e.g. because of more interference during the preamble) and then a second frame arrives, the preamble detection threshold is not relevant anymore. Because of the inability of the receiver to suppress the signal that arrived earlier, reception of the (stronger) signal that arrived second is an even more difficult task and the required SNR is even higher.

From this we can conclude that depending on the current state of the receiver and also the current state of the interference, there can be multiple SNR thresholds that a newly arriving frame must exceed for the receiver to be able to start decoding it. These are basically three relevant capture thresholds:

- *The 'normal' reception treshold.* This is the required SNR of a frame that experi-

ences no interference, just thermal noise and noise from sources other than 802.11 nodes also competing for the medium. This threshold is around 0 dB - if a frame is stronger than the thermal noise the receiver can at least start trying to decode it. Note that usually an SNR of 0 dB will not result in correct reception.

- *The 'locked' capture threshold.* This is the required SNR of a frame that the receiver needs if it is already locked to another frame and is receiving that one. Based on [3, 6] and the results in Figure 4.1 this threshold is estimated at 8 dB.
- *The 'garbled' capture threshold.* This is the required SNR of a frame that the receiver needs if it is experiencing interference from another frame but is not locked to this interference. This makes noise suppression of that other frame impossible and the required SNR much higher - based on the figure this threshold is estimated at 16 dB.

Please note that these thresholds could not be inferred from receiver circuitry, hardware configurations or other concrete data - simply because that data is not at our disposal. Instead, these thresholds are based on experiments and observed behavior, and could thus be altered slightly if necessary. Also, it should be kept in mind that this threshold does not guarantee successful reception, these are only the thresholds that a receiver needs during the preamble to lock to a stronger frame. Whether or not the frame can be received successfully can only be determined afterwards, when the bitrate of the data part of the frame and other interfering sources are known. How this is calculated is discussed in Chapter 2.4.

4.3 Simulation of Frame Capture

The behavior observed with Frame Capture, however at least reasonably documented for various communication systems (802.11a in particular), is not a standard feature in many wireless network simulators. Simulations demonstrating Frame Capture have been done, but mostly with infrastructure networks (no mobile, ad-hoc or vehicular variants) and not all network simulators currently support Frame Capture, or it is implemented in a way that does not reflect all possible scenarios identified [47, 3]. There have been efforts to implement Frame Capture in ns-2 and QualNet [47], but for our simulator of choice, OMNeT++, nothing has been done that we are aware of. In ns-3, the successor of ns-2, no Frame Capture implementation has been made up to now either.

Also, regarding the calculations of bit errors in an OFDM channel (see Chapter 2.4), not all simulators have accurate methods of calculating them. A small overview of calculation methods in various simulators is given in Table 4.2.

To summarize shortly: most simulators do have reasonably accurate methods for BER calculations, but only in AWGN channels. Considering Frame Capture, almost all simulators do not have an implementation that covers all possible Frame Capture scenar-

ios.

4.4 Potential impact on vehicular networks

VANETs are networks with an extremely high degree of mobility. The type of traffic in a VANET will largely depend on the implementations of safety applications on a higher layer, but we suspect that a large portion of the traffic in a VANET will be broadcast traffic, because independent of the application it is reasonable to assume that safety information should be disseminated as much as possible over the network (sometimes in one direction, sometimes in both) and broadcast traffic achieves this information dissemination a lot faster than unicast traffic ([49] for example proposes a smart network flooding mechanism to distribute traffic information).

It is this broadcast traffic scenario that we are interested in. Because of the design of the 802.11a & p MAC layers, broadcast information is not transmitted using the RTS/CTS mechanism and no acknowledgment is sent after reception. Also, depending on the type of information transmitted and the safety-application at a higher layer, these broadcast messages may be retransmitted for dissemination beyond the original transmitter's range. If the real-world congestion increases (more cars) the same might also happen with the network congestion (more retransmitted broadcast frames), up to a point where collisions occur on a regular basis. At this high network load scenario it is very interesting to know if Frame Capture has a significant effect on network throughput.

It has been shown that throughput can increase up to 430 % compared to the worst performing implementation [5], and they have only simulated infrastructure wireless networks. We thus suspect that Frame Capture throughput gain also be very beneficial to vehicular networks, due to the increased amount of broadcast traffic.

Table 4.2: *Bit Error Rate calculation methods in various simulators*

Simulator	FC	Method	OFDM specific	Potential quality
ns-2	no	simple SNR threshold method, later AWGN-based theoretical calculations	yes	The SNR threshold version is poor, this method in no way account for various channel conditions, modulation types used etcetera. The advantages and disadvantages of our own implementation are described in Chapter 2.4.
Jist/SWANS	no	BER lookup table. Implements functions that use a table where SNR-BER value pairs can be stored and looked up during simulations.	no	This can provide realistic results, however the actual table could not be found, only the implementation that should be able to look up values in it. An advantage of this implementation BER tables could be easily swapped, enabling realistic computations for other physical layers as well.
QualNet	no	unknown	unknown	Since QualNet is a commercial simulator, no documentation is available for free. It is unknown how this simulator calculates bit errors for various 802.11 networks.
ns-3	no	AWGN-based theoretical calculations	yes	This simulator has classes that perform the same calculations that we do, based on [48]. This same code has also been incorporated into ns-2.
OMNeT++	no	Theoretical calculations based on 802.11b	no	The bit error calculation in OMNeT is currently based on an 802.11b physical layer, which is a wideband communication system called Direct Sequence Spread Spectrum (DSSS). These calculations are not suited to OFDM or 802.11p. More about DSSS can be found in [1].

5

OMNeT++ and MiXiM

This chapter discusses the simulation framework that we use at our research group and the most important requirements of a wireless network simulator. We then briefly show how our simulation framework fulfills these requirements. This overview is not covering all aspects of wireless network simulation however, for more details we refer to the relevant documentation: [50, 9, 51, 8, 10].

5.1 Discrete-event network simulation

Most wireless network simulators are so-called *discrete-event* network simulators. Everything that occurs in the real world (or inside the hardware) is modeled as an event. An (obvious) example of an event could be that a node sends a frame. A lot of events cause new events to be generated as well - in the previous example there would be quite quickly afterwards an event that another node receives (and processes) that frame. This sending and receiving itself can even be modeled as seperate events, i.e. 'node receives frame header', 'node receives preamble' etcetera. Every event has a *timestamp*, or a moment in time that the event occurs, and at every event the node executes some code to handle that event.

Usually computer networks are very suited to be modeled with discrete-event modulators - every action at a node (e.g. a transmission, a queue drop, switching to receiving mode, changing the contention window) is modeled as an event. The simulator maintains an event list containing all the events that are scheduled by all nodes, and makes sure every node processes the generated events at the moment they are supposed to. Examples of

discrete-event network simulators are ns-2 and ns-3 [52, 53] and OMNeT++ [9]. For more information about discrete-event simulation readers are referred to [54].

5.2 Requirements of a wireless simulation environment

One can imagine that simulating a wireless network is fundamentally different from a wired network. This is mostly because of the nature of the wireless medium. This section discusses the main differences when performing wireless simulations.

5.2.1 Environment

The first main difference is the channel. Because messages are transmitted as electromagnetic waves through the air, a signal spreads in all directions and occupies a certain space for the duration of the signal. All other nodes/entities close to the transmitter will hear the same message, and depending on the used frequency and modulation these other nodes cannot transmit at the same time. This requires fundamentally different access rules at the MAC layer. On the physical layer, propagation of a signal through the wireless medium is very different from propagation through a wire and a receiver has to deal with a lot more disturbances and noise in the signal.

Simulating this wireless medium is not an easy task. Radio waves can behave very unpredictably, they can bounce off objects and penetrate through walls, creating many channel effects like shadowing, scattering, diffraction, multipath propagation and short- and long-term fading (for more info on these channel effects see Appendix A). Also, because of the shared medium, nodes might accidentally transmit simultaneously, creating extra noise or even disrupting each other's communications. All these effects are different from the typical attenuation in a copper wire or glass fiber, and should be taken into consideration in the simulator. Obviously the standard wired 'connections' from OMNeT++ cannot be used; the wireless medium is too important to abstract from it.

5.2.2 Node movement

Inherent with the channel is the increased movement of the nodes. In a wired network the location of a node (w.r.t. another node) is not important; the signal must still travel the same distance through the wire. In a wireless network, the distance is of great importance, it determines the strength of the signal and thus the successful reception probability. The location of a node in relation to the environment is also important. If there are objects blocking or reflecting the signal, that will also influence the signal strength at the receiver. There are wireless network scenarios where the nodes have a fixed location, so signal strength from one node to another remains constant. In a

vehicular network this is not usually the case, nodes can all move at significant speeds and distances between nodes change rapidly.

Node movement implies that a simulator should not just logically connect nodes (through 'wires' in the simulator), but maintain a very detailed view of the simulation space, the location of the nodes, their directions and movement patterns and the locations of objects in that same space. All this information should be readily available for any node if it wants to transmit a frame through the air. The simulator should then know which other nodes will be reached by the transmission, and also which nodes will not be able to receive it but possibly suffer from the interference. Locations of objects should be managed, and all these factors (distance, propagation, channel effects) should be modeled as good as possible without causing too much computational overhead.

5.3 OMNeT++

OMNeT++ is a discrete event simulation environment. It is entirely built in C++ and has a very modular structure. It is itself a very basic simulator (i.e. also usable for simulations outside of the wireless domain), but it also provides the tools for developers to write their own extensions on top of it. Programmers can write modules that represent nodes in a network for example, which can then be used to simulate all kind of network scenarios. It uses a high-level language to define smaller and larger components (e.g. a network interface card) built of smaller parts representing various network layers). All kind of protocols, both wired and wireless, can be simulated with OMNeT++. It provides various types of interfaces to visualize simulations, structures to exchange information or monitor certain parts of the simulation etc.

OMNeT++ has two different types of building blocks: simple and compound modules. A simple module is a basic functionality block, which can be written in either C++ or OMNeT++'s native (NED) language. A module has input and output gates, through which it can communicate with other modules. These other modules can be 'internal' (e.g. between two network layers in a wireless node), or 'external' (like two routers that connect over an Ethernet cable). Between two gates a connection can be defined with parameters such as a data rate and bit error rate. If a certain connection is used a lot, a connection type (or channel) can be defined to reuse the connection in multiple places.

Compound modules can be formed connecting one or more simple modules through their gates, creating a 'cardboard box' around the simple modules. Compound modules can then again be used within other modules, in this way one can create a nested hierarchy, increasing functionality and complexity step by step. The connections created between gates can be used to exchange messages. Messages can contain arbitrary data structures, they can represent packets in a computer network, but also jobs or customers in a queuing network, depending on the type of simulation.

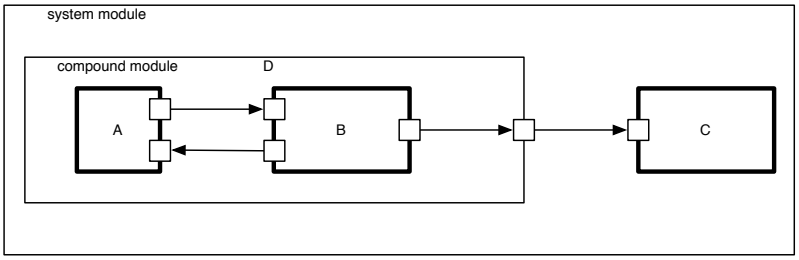


Figure 5.1: *OMNeT++ module structure (Based on [55]).*

In Figure 5.1 the OMNeT modular structure is further clarified visually. In the example A, B and C are simple modules and A and B are used together in a compound module called D. A and B have in- and output gates through which they are connected to each other. D only has an output gate to C and is thus unable to receive input.

5.4 MiXiM

MiXiM is an extension of OMNeT++ specifically created to simulate all forms of wireless communication, and take into account all the extra requirements discussed above. It provides a few base models that tailor basic OMNeT++ functionality to the needs of the wireless medium, and is just as OMNeT++ highly modular and extensible; if there is a wireless protocol that is not yet implemented, most of the building blocks are there to implement it ourselves. Below, we will discuss how MiXiM covers the basics of wireless simulations.

5.4.1 Environmental models

The environment of the simulation (in MiXiM also referred to as the *playground*) is the space (either 2 or 3-dimensional) where objects and nodes are placed. *Nodes* are the communicating entities of the simulation, *Objects* are non-communicating entities that can possibly affect signal propagation through the wireless medium. Nodes are dimensionless, they have a position, and can be moving according to a certain movement pattern. Objects also have a dimension, a position, possibly some frequency-dependent attenuation factors and can also be moving. MiXiM manages the playground and all entities on it with an *ObjectManager* and a *ConnectionManager*. The *ObjectManager* manages the objects and their locations in the simulation environment, and can be queried to see if a certain connection between two nodes is hindered by an object (which would decrease the signal strength if the two nodes were communicating). The *ConnectionManager* maintains and updates connectivity information, i.e. which nodes are currently within each others interference range. In MiXiM, a connection between two nodes exists if they are within the maximum interference distance of each other; but

this does not mean that they are able to communicate. However all nodes that are ***not*** connected, definitely do not interfere with each other. Following this principle, if a node wants to receive a message from a neighboring node, it will also receive all interfering signals and can decide based on the interference and SNR if the received message can be received correctly or not.

This approach ensures that all relevant frames arrive at the receiver, without creating too much overhead. If e.g. all nodes were connected and the receiver needs to find out for every frame sent in the simulation space if it is strong enough or not, a lot of unnecessary frames would be delivered at nodes that are very far away from the transmitter.

5.4.2 Wireless channel models

A very important factor in simulating wireless communication is the wireless channel. A wireless medium exhibits properties and effects very different from a wired medium which are crucial to the performance of the system. To account for this, MiXiM incorporates a few channel models that simulate path loss, multipath-, shadow- and other types of fading, various types of interference etc. When expressing these propagation effects in the channel models, the exact signal behavior is abstracted out. However, the different effects can all be calculated separately, and applied to the signal if deemed necessary. In that way the accuracy of the channel modeling and computational load can be adjusted easily. The channel effects are modeled as factors of the instantaneous SNR of the received signal.

5.4.3 Physical layer

The physical layer of the wireless simulations (and how it is implemented in MiXiM) is by far the most important for our research, since Frame Capture and all the other channel effects we described occur at this layer. In this section we will discuss how the physical layer is simulated, and how this is implemented in MiXiM. Note that this section describes the current, unmodified implementation of the MiXiM physical layer, without the modifications required to enable Frame Capture.

The physical layer in MiXiM is responsible for sending and receiving frames, applying the various channel effects, collision detection and bit error calculation. Different responsibilities are handled by different classes, as can be seen in Figure 5.2. The *BasePhyLayer* is the most important class; it carries out basic functions such as message handling from the MAC layer and connects/calls other functions and classes. The *AirFrame* class is the container class for a *Signal*; it is the 'message' that is exchanged via the in- and output gates within the simulator. The *AirFrame* contains information like the start time of the *Signal* (at what point in time the transmission started).

The *ChannelInfo* class is responsible for managing the current status of the channel,

specific to the node. At any moment *ChannelInfo* knows which *AirFrame*s are currently being transmitted and are currently in the air around the node. In the simulator, *AirFrame*s arrive instantly, but a real signal has a duration in which it occupies the medium. This problem is also partly solved by *ChannelInfo*; it knows about the *AirFrame*s that are currently being received, but also of the *AirFrame*s that are currently interfering with an ongoing reception, and of the *AirFrame*s that might have already ended, but interfered with (some part of) an *AirFrame* that is currently being received. The problem of instantaneous reception is also solved by the *Decider*, more about that can be read below.

The *Signal* object is the abstract version of the signal in the wireless medium. The channel effects discussed earlier are contained in so-called *AnalogueModel*s, they create filters (or *Mapping*s) which are applied to the *Signal*, simulating attenuation of a real signal. There are various kinds of *AnalogueModel*s:

- *RadioStateAnalogueModel* This *AnalogueModel* simulates the effect of the radio state on the reception of the signal. If the radio is not in receiving state, for example, nothing will be received and the 'attenuation' will be 100 %.
- *SimplePathLossModel* A simple implementation by MiXiM adding a path loss filter to the signal.
- *LogNormalShadowing* A model that describes shadowing using a log-normal distribution (also see Appendix A).
- *JakesFading* An implementation of the Rayleigh fading model based on summing sinusoids.

An important part of the physical layer is the *Decider*, it processes every incoming *AirFrame*. At the moment reception starts, the *Decider* checks if this *AirFrame*'s signal is strong enough to be received or not. It does so based on the information in the *ChannelInfo* (which other frames are currently on the air), and based on the current state of the receiver; whether it is already receiving another frame or not. After reception of the frame has ended, the *Decider* determines whether the frame was received correctly or not, based on the signal-to-noise ratio of the frame (in this context the other frames on the medium at the same time are considered noise too) and based on the bit error calculation functions, which indicate what the probability of successful reception is based on the signal-to-noise ratio. If the data rate was too high or the SNR too low, the signal has a high probability of being received with errors. The *Decider* can also request the frame from the physical layer for second inspection (for example after the header has been received), and it pushes the packet encapsulated in the *Signal* to the MAC layer if it is received correctly [10].

The *Decider* is one of the most important parts we need to adapt in order to implement Frame Capture in the simulator. This is because the *Decider* processes every frame upon reception, and within the *Decider* we can build logic that checks the signal strength of newly arriving frames, compare it to the signal strength of other frames and if necessary,

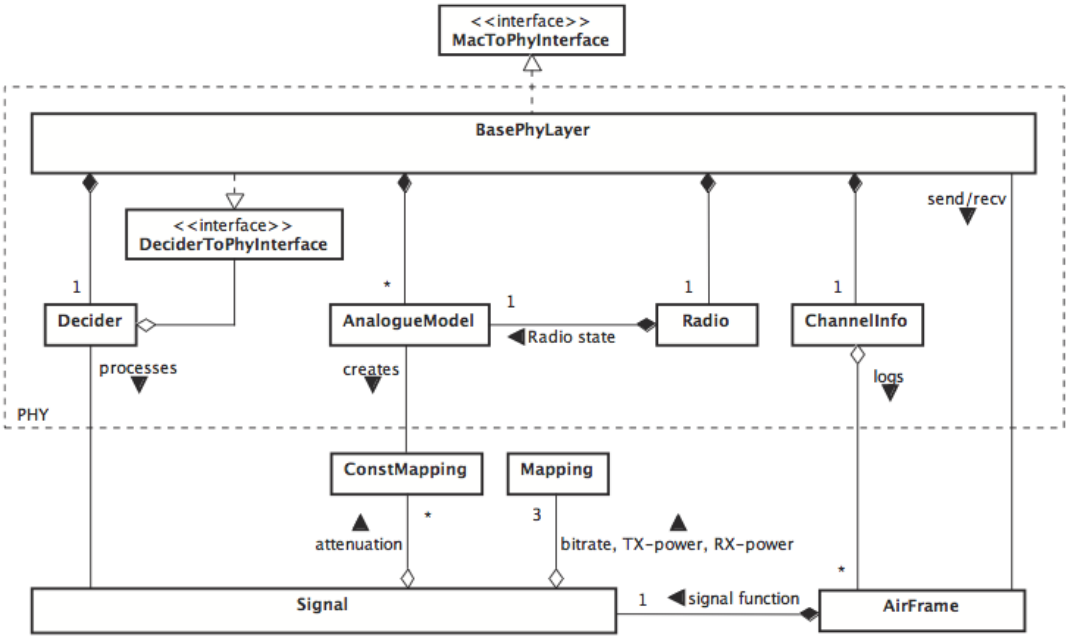


Figure 5.2: *MiXiM physical layer functionality diagram (Copied from [10], Figure 2)*

make a *capture decision*, where the *Decider* lock to the new frame and discards another one, even if that one was currently being received.

The relation between the various components becomes clear when looking at Figure 5.2. The *BasePhyLayer* sends and receives *AirFrame*s (which are passed to it from the MAC layer). *ChannelInfo* maintains an up-to-date view of the current medium and logs the *AirFrame*s. One or more *AnalogueModel*s are applied to the *Signal* that affect the perceived SNR. The *Decider* processes every *AirFrame*, and calculates based on the SNR the probability of successful reception.

To further clarify the inner workings of the physical layer, Figure 5.3 shows the chain of events when transmitting. When a frame arrives from the upper layer for transmission, the 802.11 MAC layer performs the basic processing. It creates the *Signal* object with some control information already filled, attaches this to a *Mac80211Pkt* which contains the data to be sent, and performs other basic tasks like encapsulation. Then the physical layer creates an *AirFrame*, the container class for the data that is being sent and the *Signal*. Note that the data is not contained in the *Signal*, the *Signal* is merely a class defining the physical parts (bitrate, transmit power, start time duration of the frame), without the actual data that is being transmitted. The signal is contained within the *AirFrame*.

Then the *ChannelAccess* class asks the *ConnectionManager* for a list of nodes connected to this node. The ConnectionManager gives a result based on the locations of the other

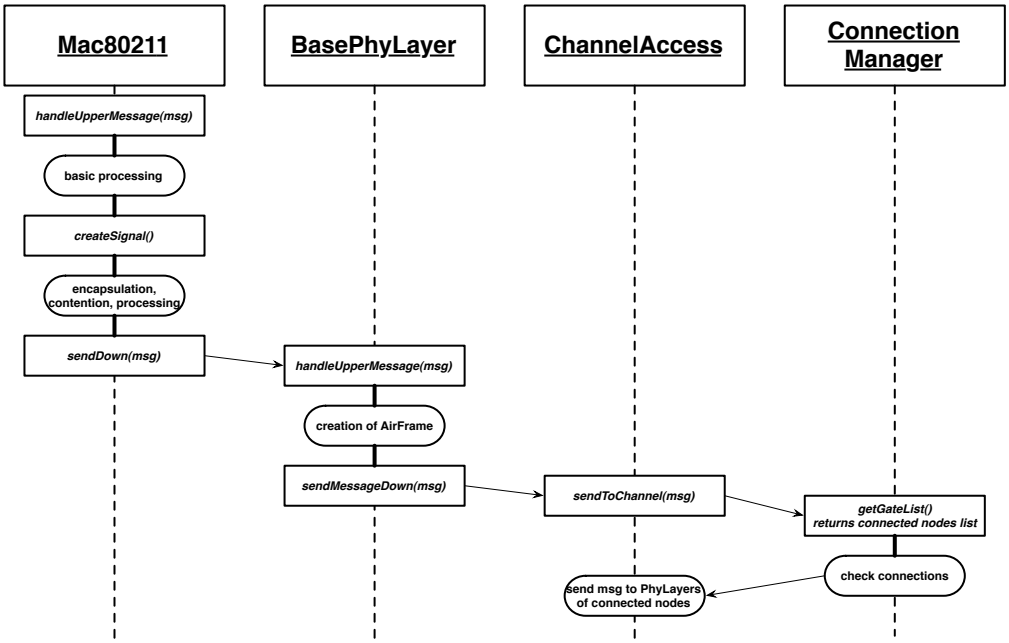


Figure 5.3: *MiXiM physical layer transmission flow diagram*

nodes and objects. *ChannelAccess* then delivers the *AirFrame* to the input gates of the physical layers of other nodes. If propagation delay is also simulated, *ChannelAccess* calculates the propagation delay based on the distance between the nodes and calls the *sendDelayed()* method, causing OMNeT++ to run a self-timer delivering the *AirFrame* at the input gate at a later moment in the simulation.

Reception is a bit more complicated; when a message arrives at an input gate, many more steps must be performed. The functionality blocks in Figure 5.4 show the various steps that the receiver might take. Both figures are colorized to see more clearly what happens in which step.

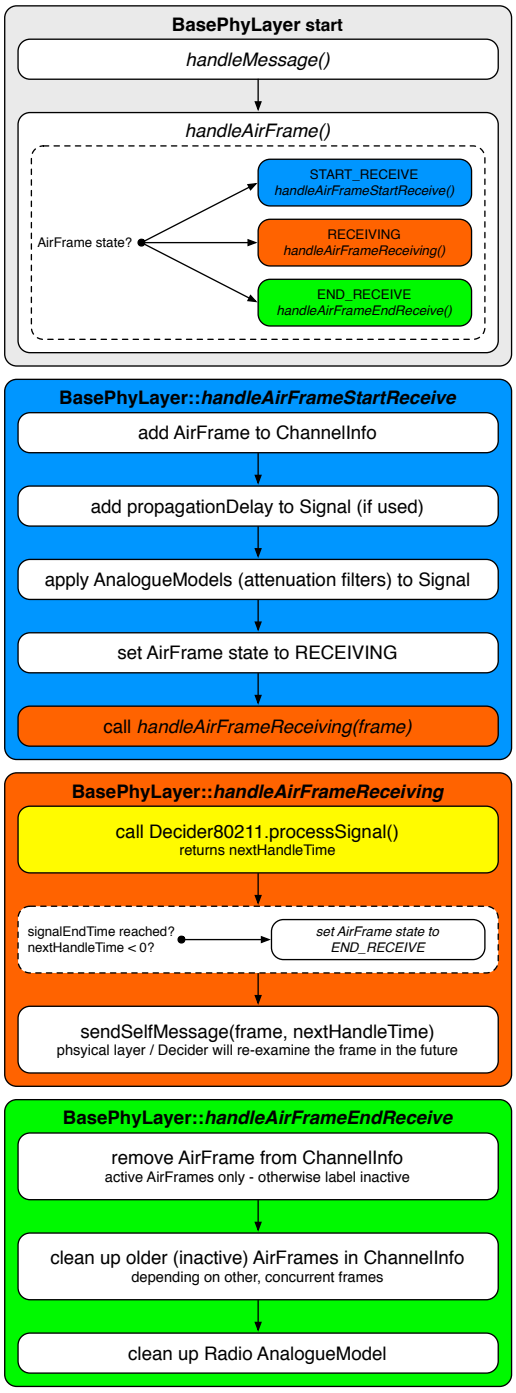
When a message (an *AirFrame*) is received, it is first handled by the *BasePhyLayer*, which calls the method *handleAirFrame()*. Depending on the state of the *AirFrame*, it is then passed on to another method. Note that an *AirFrame* can be processed multiple times, so that the receiver does not have to decide in a single moment what to do with the frame. In the current implementation the frame is handled by the *BasePhyLayer* at least twice, once when reception starts and the second time when it ends. As mentioned above, we need to expand this behavior in the *Decider* to enable capturing frames during or after the preamble.

The first time the frame is handled the *BasePhyLayer* adds the *AirFrame* to the *ChannelInfo* and applies the attenuation filters in the *AnalogueModel*s to the *Signal*. After that, the frame is passed to the *Decider*. If the frame arrives the second time at the

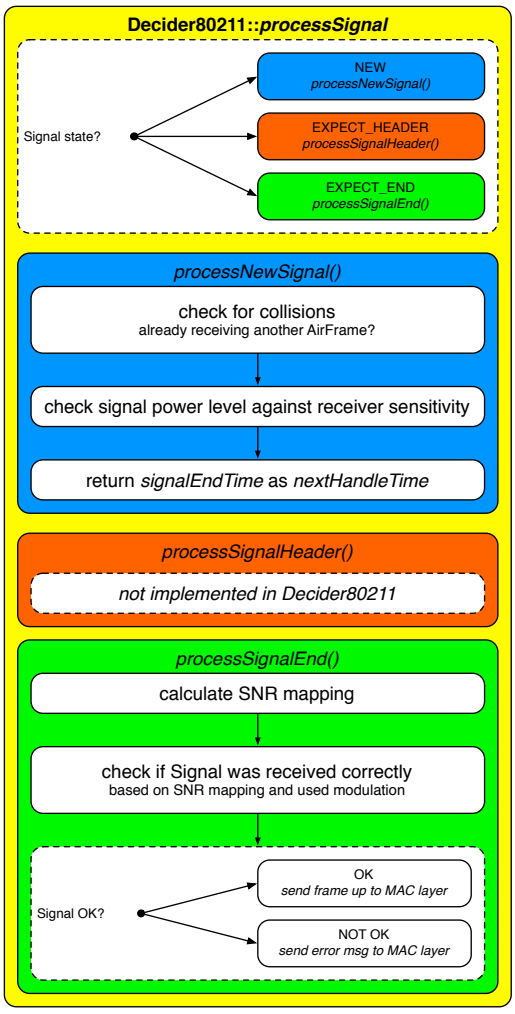
BasePhyLayer, it is passed to the *Decider* immediately.

The *Decider* then checks for itself if it has seen this frame earlier or not. The current *Decider* implementation has only two main moments when it expects to see the *AirFrame* and the *Signal* in it: when the frame arrives and when it ends (i.e. when reception has been completed). In the current implementation, this means there are two methods: *processNewSignal()* and *processSignalEnd()*. There is also a function to process the frame after the header (*processSignalHeader()*), but that one is not being used. First, the *Decider* checks for collisions and checks the signal power level to see if the signal is even strong enough to be received. Then it returns the next time in the simulation it wants to receive the frame for second inspection. If this time is smaller than zero, it means that the frame can be discarded directly. Other times mean that the *BasePhyLayer* and *Decider* will analyze the frame again at a later moment in time. The current implementation returns the signal end time (the time when reception has been completed) as the time it wants to handle the frame again.

At a later point in time, when *processSignalEnd()* is called, it determines definitely if the frame has been received correctly or not. It computes an SNR mapping based on the *AirFrame*s present in *ChannelInfo* (which it requests through *BasePhyLayer*), the receiving power of the *Signal* and other noise factors, and decides based on the SNR during the frame reception if the frame has been received correctly or would have contained errors. If it has been received correctly, the *Decider* directly passes the frame upwards to the MAC layer, if not, it sends a control message indicating an error.



(a) BasePhyLayer



(b) Decider

Figure 5.4: *MiXiM reception functionality blocks. The colors of the blocks indicate which functionality is called at which moment.*

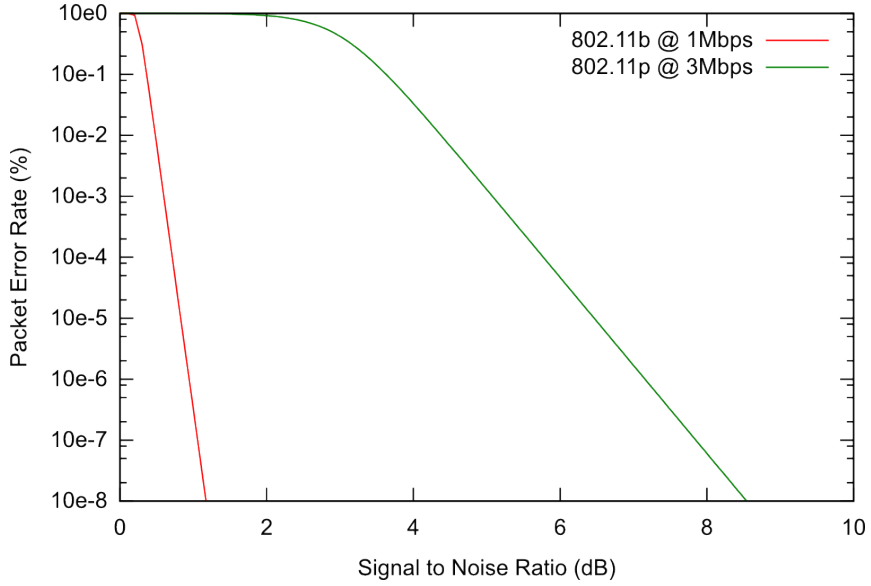


Figure 5.5: *802.11b and 802.11p PER curves at their lowest bit rates (1 and 3Mbps)*

Bit error calculations in the current physical layer

In the current *Decider* implementation the bit error probabilities are based on the 802.11b physical layer, and calculated with the formulas given in Equation 5.1. In these equations BER is the bit error rate, Packet Error Rate (PER) is the probability that the entire frame has errors, and L is the length of the frame in bits. Note that these equations are for the lowest bitrate (1 Mbps). What the precise calculations are and how they are derived is not relevant to this thesis, it is only important that they cannot be used for our vehicular network simulations, simply because the physical layers are too different.

$$\text{BER} = 0.5 \cdot e^{\frac{-\text{SNR} \cdot 20 \cdot e^6}{1 \cdot e^6}} \tag{5.1}$$

$$\text{PER} = (1 - \text{BER})^L \tag{5.2}$$

Figure 5.5 shows the packet error rates for 802.11p and 802.11b at their lowest bitrates, as calculated by the MiXiM, for frames that are 312 bits long.The 802.11b calculations were already present, the 802.11p calculations have been added by us (see also the Chapter 6 regarding the implementation). As shown, 802.11b gives a bit error probability that is a lot lower and it falls a lot sharper than its 802.11p counterpart (note the logarithmic scale on the y-axis).

The effects of coding, forward error correction and modulation as they are used in 802.11a or 802.11p are not yet implemented in the simulator. The bit error rate calculation in

the current *Decider* implementation should, as mentioned above, be updated. How this is done is discussed in the next chapter.

6 Implementation

In Chapter 5.4.3 we have seen that the physical layer of MiXiM is already quite advanced. However some crucial parts are not yet tailored to 802.11a or 802.11p. This chapter discusses the changes we made to the physical layer. With these changes, we will be able to simulate vehicular networks with a much greater degree of accuracy than the current implementation.

6.1 Frame Capture Implementation

According to the simulator's physical layer behavior described in Chapter 5.4, every frame is analyzed by the Decider when it arrives. The Decider then looks at the channel, the received frame's SNR and gives back a time where it wants to look at the frame again. The Decider also always manages the current frame (i.e. the frame that the receiver is currently locked onto).

What we needed to do was expand the Decider's behavior to account for Frame Capture. When studying the current implementation, we found out that it 'by accident' implemented some form of capture - if a frame arrived first and the receiver could lock onto it, any other frame arriving during that frame's ongoing reception would be ignored. This partly like the Prism chipset that can only capture stronger frames when they arrive first, or during the preamble. However, the current implementation could only capture frames that arrived earlier, since their was no collision detection built in at all. We upgraded that behavior to also account for frames arriving during each others preamble, and we included Atheros behavior, as described in Chapter 4. The new Decider needs

to make a decision to lock onto a new frame, based on the current and the new frame's signal-to-noise ratios.

Receiver behavior is already described in Chapter 5, here below we describe the additional required behavior that is related to Frame Capture. The reception behavior that we have implemented is as follows:

- If a new frame arrives, the Decider temporarily stores that frame but makes no decisions about it yet. Instead, it waits until the preamble of the frame has passed completely.
- After the preamble, the Decider makes the decision whether this frame should be received or not. First, it decides on which capture threshold it needs to use (based on the thresholds described in Chapter 4.2)

  • if there is no other frame being received right now and there was no interference during the preamble, the normal reception threshold is used (set by us at 0 dB).

  • if there is no other frame being received right now but there was interference during the preamble, the 'locked' capture threshold is used (set by us at 8 dB). This is because the preamble is the most important part of the reception process; if there was interference during the preamble (e.g. a frame ends exactly during the preamble of another frame), the locking process described in Chapter 2.3.3 is disturbed by the other signal and the probability of the receiver correctly synchronizing to the sender has decreased.

  • if the receiver is currently receiving another frame, the 'locked' capture threshold is also used (8 dB).

  • if the receiver is currently suffering from interference (i.e. another frame is in the air and the receiver is not locked onto it, the 'garbled' capture threshold is used (set by us at 16 dB).
- Then, if the frame's SNR exceeds the required threshold, the Decider makes the decision to lock to this frame and start receiving it. If the SNR is not high enough, the frame is discarded (and seen as interference for the rest of the duration of the frame). The Decider then requests to see the frame again after reception is complete.
- Then, when the frame is received completely, the SNR of the frame is computed as usual (dividing the frame's signal strength by the cumulative signal strength of all the interfering signals and the thermal noise). Then for that SNR and looking at the data rate (and thus modulation type used), the Bit Error Rate calculation formulas are applied to calculate the probability that the frame has bit errors. These BER and PER formulas predict the *probability* that a frame has been received correctly, the simulator then decides whether the frame has indeed been received without errors or not. If, for example, the PER formula predicts an

80% correct reception probability, the simulator draws a random number between 0 and 1 using a (pseudo-)random number generator. If the drawn number is below 0.8, the frame has been received correctly, if it is between 0.8 and 1, the frame is considered lost.

Note that the last point mentioned is already present in the current Decider implementation, but the Bit Error Rate calculation formulas that are available are specific to 802.11b. We needed to update the implementation to account for 802.11p, using the results from [20] as described in Chapter 2.4. How we implemented that part is described below, in Chapter 6.2.

Basically all the behavior described above is implemented in three *Decider* functions that process the frame at different points in time during its reception. These functions are *processNewSignal()*, *processPreamble()* and *processSignalEnd()*. The enhanced *Decider* behavior is visualized in Figure 6.1. This does not constitute the complete Decider behavior, the parts that have not changed have been omitted and can be viewed in Figure 5.4. Also note that the behavior is slightly simplified in the diagram; if after the preamble there is more than 1 frame being received (e.g. one being received correctly and one as interference), the required threshold is always 16 dB because the receiver cannot possibly be locked to all frames. Also, in the real implementation, various other functions (such as collection of metrics) are implemented but this behavior is not reflected in the diagram.

6.1.1 Justification of capture thresholds

The capture thresholds are mentioned in various sources [15, 3]. Lee et. al. [3] derived its value through simulations, indicating that for Atheros chipsets it is possible that 8-10 dB is simply the minimum needed to decode a preamble in the presence of interference or that this threshold is set to 8-10 dB to protect the previous frame and not switch frames too fast. This means that this threshold might be configurable, but we have found no implementations of Atheros firmware where it is. It can also be that there are automatic noise suppression circuits (as also described in Chapter 4.1.4 and [14]) in place, but there are no detailed circuit diagrams of any 802.11 chipset available and understanding them usually requires a degree in electrical engineering. Regarding the Atheros chipsets with the most advanced capture behavior; Atheros does not supply data sheets with exact information on this matter, and any requests for more information were not answered. Also, in the reverse-engineering of the open-source Atheros firmware (known as OpenHAL, [56]), there was no information that pointed to Frame Capture or that pointed to configurability. Based on the experiments that have been performed and the literature we can assume that for 802.11a and 802.11p this value is reasonably accurate, for other wireless technologies it might very well be different.

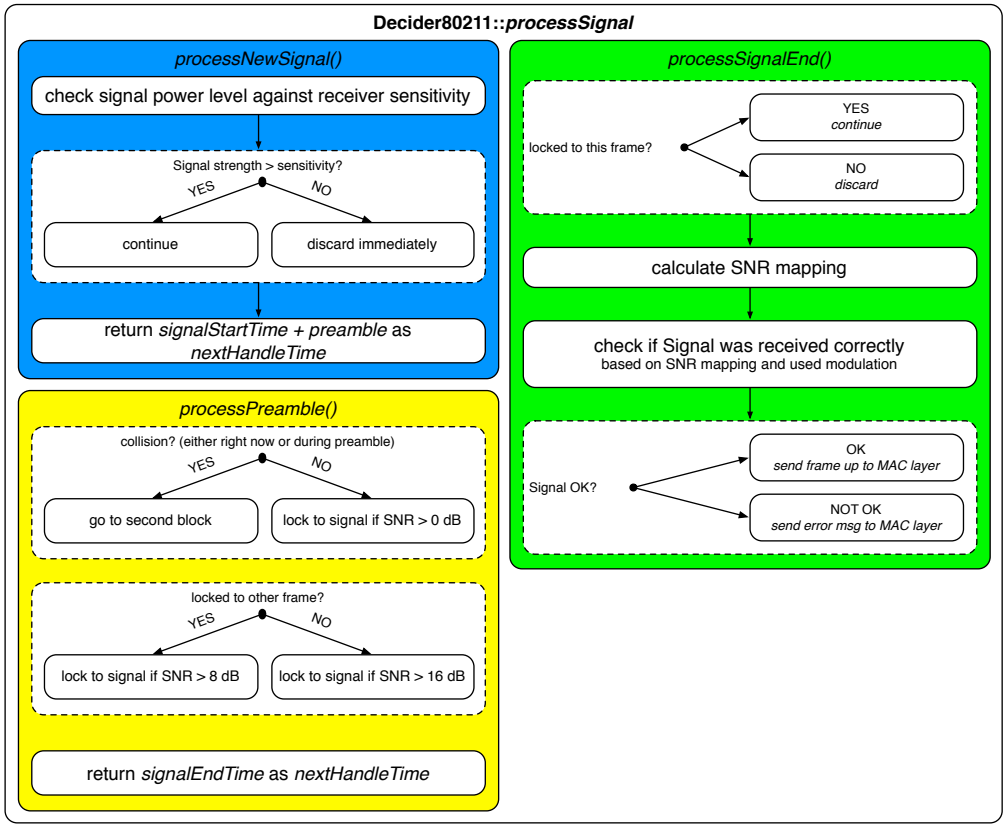


Figure 6.1: *MiXiM Decider - enhancements made to enable Frame Capture*

6.2 Bit Error Rate Calculation Implementation

The theory behind calculating bit errors is discussed in Chapter 2.4. However, the calculations necessary to predict accurately if a frame arrives correctly or not are quite complex and computationally intensive. If these calculations must be performed for every packet that is being received, simulation time and cost would rise significantly. As a solution to that problem, we have decided not to implement these calculations directly, but work with a derivative of them. In [20] and from that, Figure 2.13, the packet error probabilities for all the 802.11p data rates are shown (every data rate corresponds to one modulation type and coding rate). Instead of performing all the calculations needed to calculate these values, we tried to find a simple mathematical function that resembles the PER curves as closely as possible. We do this with a method called *curve fitting*; finding the best matching function for a series of data points. In our case, the data points are extracted from Figure 2.13. We used the first line from 2.13, corresponding to the lowest bitrate (3 Mbps) and a frame length of 312 bits. The same method is used in [57], although for a different (HSDPA) physical layer. Based on the fitting function from that paper (given in Equation 6.1), it was a pretty straightforward task to use this method

to fit our PER function. In [57], the CQI is the Channel Quality Indicator, which is a constant. We can rewrite $\text{BLER}^{-0.7}$ and reduce the complexity of the function (and add our own constants) to Equation 6.2.

$$\text{SNR} = \frac{\sqrt{3} - \log_{10}(\text{CQI})}{2} \cdot \log_{10}(\text{BLER}^{-0.7} - 1) + 1.03 \cdot \text{CQI} - 17.3 \tag{6.1}$$

$$\text{SNR} = x \cdot \log_{10}(\text{PER}^{-z} - 1) + y \tag{6.2}$$

Then, by playing with x, y and z we fitted the function as closely as possible to the first curve in Figure 2.13. We found x to be $\sqrt{3}/2$, y to be 3.5 and z to be 0.8 for SNR values higher than 3. However, in [57] the axis are swapped (compared to Figure 2.13), so we also needed to solve the equation for SNR, to make it the variable instead of the result. If we first fill in the values for x, y and z, we get

$$\text{SNR} = \frac{\sqrt{3}}{2} \cdot \log_{10}(\text{PER}^{-0.8} - 1) + 3.5 \tag{6.3}$$

We can then substitute $\text{PER}^{-0.8}$ with x (note that x is a different variable now) and we get

$$\text{SNR} = \frac{\sqrt{3}}{2} \cdot \log_{10}(x - 1) + 3 \tag{6.4}$$

We can start rewriting this with the steps shown in Equation 6.5 and the subsequent equations. In the second step we bring everything except the $\log_{10}(x-1)$ to the left side of the equation, we then tidy up the left part and in the fourth step we rewrite $y = \log_{10}(x-1)$ to $x = 10^y + 1$.

$$\text{SNR} = \frac{\sqrt{3}}{2} \cdot \log_{10}(x - 1) + 3 \tag{6.5}$$

$$\frac{\text{SNR}}{\frac{\sqrt{3}}{2}} - 3 = \log_{10}(x - 1) \tag{6.6}$$

$$\frac{2 \cdot \text{SNR}}{\sqrt{3}} - 3 = \log_{10}(x - 1) \tag{6.7}$$

$$x = 10^{(2 \cdot \text{SNR}/\sqrt{3}) - 3} + 1 \tag{6.8}$$

We then again substitute x for $\text{PER}^{-0.8}$ and we get Equation 6.9:

$$\mathrm{PER}^{-0.8} = 10^{(2\cdot \mathrm{SNR}/\sqrt{3})-3} + 1 \tag{6.9}$$

$$\mathrm{PER} = \frac{1}{\left(10^{(2\cdot \mathrm{SNR}/\sqrt{3})-3} + 1\right)^{1.25}} \tag{6.10}$$

This function fits almost precisely with the first line (packet length of 39 bytes = 312 bits) of Figure 2.13. However to derive a function for an arbitrary packet length we must perform one more additional step, which is shown in Equation 6.11. We write the packet error rate for 312 bits back to (upper bound of) the bit error rate (details can be found in [20]), and then from that bit error rate we calculate the PER for any arbitrary packet length (in bits).

$$P_u^m = 1 - (-\mathrm{PER}_{312} + 1)^{1/312} \tag{6.11}$$

$$\mathrm{PER}_L = 1 - (1 - P_u^m)^L \tag{6.12}$$

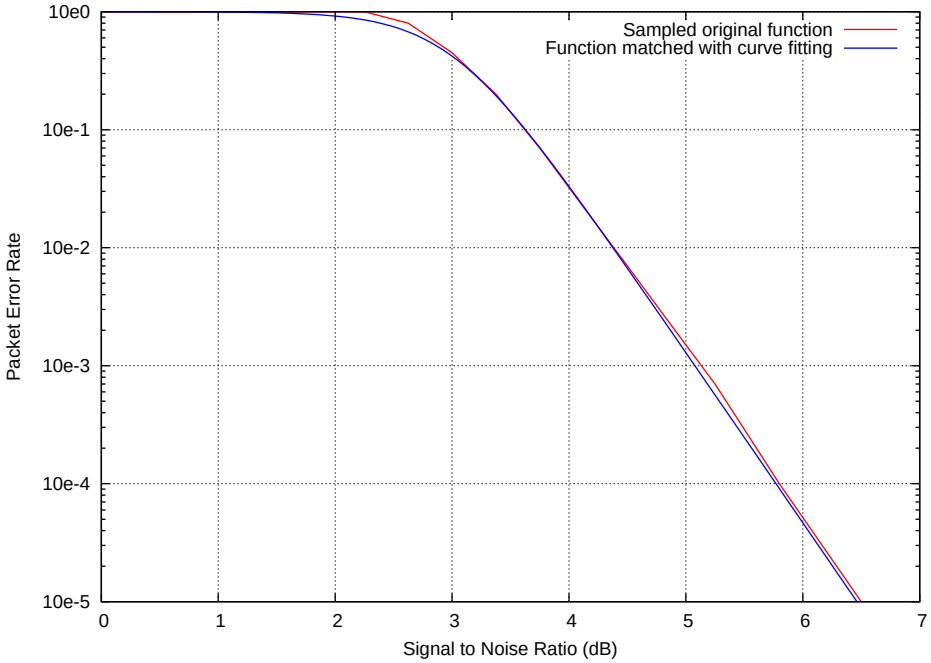


Figure 6.2: *Comparison of the approximation function with the sampled function for BPSK (code rate 1/2) from Figure 2.13.*

The results from comparing the function from Figure 2.13 for the lowest bitrate (see also [20]) and our approximation function is shown in Figure 6.2. Note that the approximation curve is a little bit below the sampled curve from the theoretical results, this

is partly because of the sampling frequency. It is hard to correctly sample data points from Figure 2.13 and therefore also not useful to validate the quality of the approximation function in another way than with the naked eye, but if we would obtain more (and more precise) data points we could calculate the surface of the difference between the two functions to obtain a measurable quality indicator of the approximation function. For our purposes however (and taking into account that these formulas are already an approximation because in the real world we do not have an AWGN channel), this approximation function fits certainly good enough.

We have performed this similar approximation for the data rates 3, 4.5 and 6 Mbps, and this can easily extended to the other data rates. As said before, the main advantage of this approach is that simulations will be a lot less expensive from a computational point of view, and that we get accurate packet error rates for various SNR ratios, at least for the AWGN channel model. For other channel models, this method could prove to be less accurate, but improving it further is left as future work.

6.3 Validation

Before we continue to simulate vehicular network scenarios and investigate the effect that Frame Capture has on VANETs, we should first attempt to validate our implementation of Frame Capture. We do this to verify that the behavior of Frame Capture in our simulator is the same as the behavior demonstrated by real chipsets. The best way to validate whether our implementation is correct is by imitating a well-described Frame Capture experiment. By creating a simulation space that looks exactly like the real experiment (in terms of number of nodes, node placement, traffic type, traffic generation rate etc.) we can compare the results of our simulated physical layer with the test results of the real experiment.

We tried to validate our implementation using the experiment described in [3]. We did this for a variety of reasons:

- The experiment described in this paper is small and easy to recreate.
- The paper focuses solely on Frame Capture.
- There are papers and experiments with more recent results, however they are from the same authors and those papers use wired topologies to simulate Frame Capture, which is itself already an abstraction from the real world [15] . Because of this abstraction the results of this work do not have as much value to validate as the experiments performed with real wireless cards. Also, the circumstances of these experiments are harder to recreate. Unfortunately there is not much literature available from other authors / research groups.

6.3.1 Setup of the experiment

This experiment (described in [3]) is set up as follows: there are 5 nodes, two senders, two sniffers and one receiver. All nodes are equipped with Atheros wireless network cards. One of the senders is designated as the 'Interferer', and the other is the 'Sender'. The sniffers are positioned very close to the senders and their only goal is to monitor and timestamp all generated traffic - they do not generate traffic themselves. The receive antennas of both senders are disabled, so they do not perform carrier sensing. This means both nodes will always think that the medium is idle and will always start transmitting as soon as they have a frame waiting for transmission. An obstacle is placed between the two senders as well to make the sniffers always receive the frames of their corresponding sender node, not the frames of the other one. The reported noise power value reported by the network cards is fixed to -90 dBm.

The Sender and Interferer then start to generate a lot of User Datagram Protocol (UDP) traffic to the Receiver, with various payload sizes (between 1000 and 1500 bytes) at random intervals. The sender also varies its transmission power so that the receiver experiences many different signal-to-noise ratios. The traffic generation rate is so high that almost all frames collide at the receiver, although not clearly defined in [3]. This means we just have to assume that they generate as many collisions as possible. Every collision creates a situation where the receiver can potentially capture a frame. With the timestamp information of the sniffers and the (also timestamped) information of which frames arrived, the researchers can figure out which frames were captured under what circumstances. The experiment setup is also shown in Figure 6.3. All capture events are classified as discussed in [3] and Chapter 4.

Unfortunately not all variables were clearly defined in the experiment (e.g. payload size). We assume that the intervals between transmissions from both senders were randomly distributed (which distribution is used exactly is not described), meaning that the interference could start at any moment during the frame's reception.

6.3.2 Difficulties

However, during the validation process quite a few difficulties arose - even before simulating we could predict that the results we would obtain with our simulation would be different from the results of the experiments themselves. There are a few important factors, mostly concerning the bit error calculations.

The results from [3] indicate (this can also be viewed in Figure 6.4) that when the stronger frame arrives first, the required SNR at 6Mbps is very close to 0 dB, i.e. both frames can be almost equal in strength and the first frame will still be received. However, with the new bit error approximation functions that we have incorporated in our physical layer implementation (stemming from [20]), receiving any frame with an SNR of less than 2.5 dB is impossible. Predicted bit error rates in an AWGN channel are always near 1 below

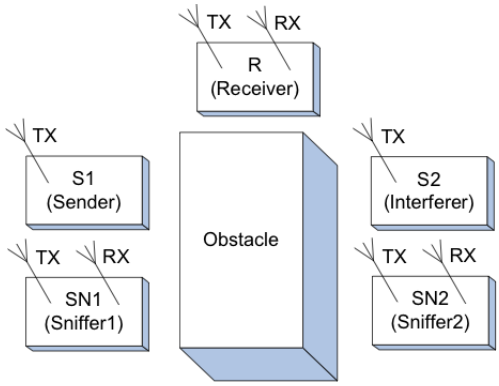


Figure 6.3: *Setup of the experiment used to validate our implementation of Frame Capture. This setup is also used in [3].*

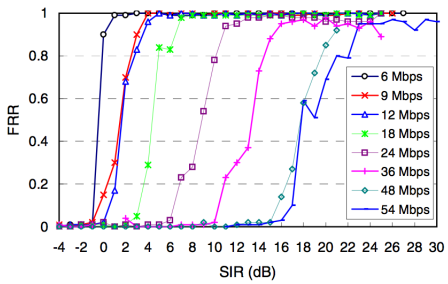
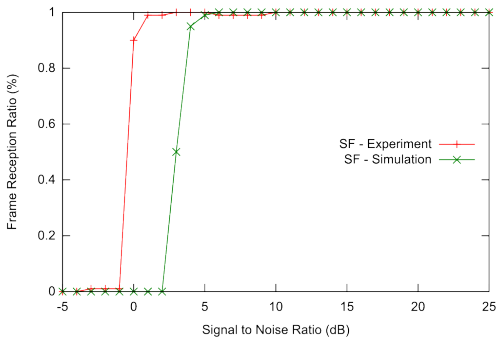


Figure 6.4: *Required SNR for Frame Capture scenarios where the stronger frame arrives first (copy of Figure 4.1, included here for readability).*

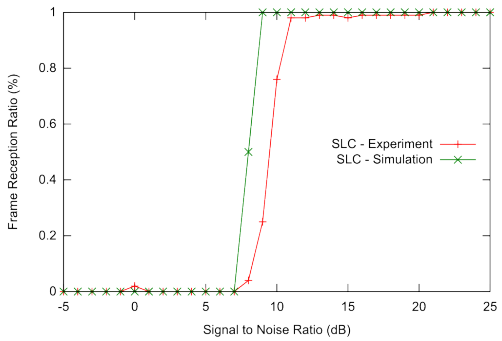
2 dB, as can also be seen in Figure 2.13 (and in Figure 5.5). If we use these bit error calculations and simulate frame capture the results cannot possibly be the same: it is impossible to get a 0.9 Frame Reception Ratio between 0 and 2 dB. As can also be seen from Figure 5.5, both the 802.11b and the 802.11p formula give a packet error rate of 1 around 0dB, this can impossibly give a Frame Reception Ratio of 0.9 as with the lowest bitrate in Figure 6.4.

On top of that is the problem that this bit error approximation is based on 802.11p, where the experiment uses 802.11a hardware. Since 802.11p is designed to be more robust than 802.11a (see also Chapter 3.4), the SNR required for successful reception should actually be lower, not higher. This theoretical difference is around 3 dB if exactly the same kind of coding is used [24].

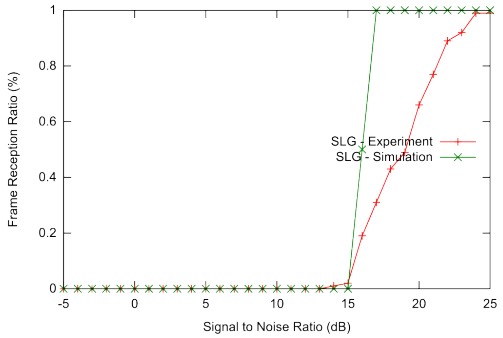
Furthermore it should be noted that the bit error approximation is based on formulas which apply to an Additive White Gaussian Noise channel model (see also Chapter 2.4). This might be a valid channel model compared to the laboratory environment used in the experiment, but on a real highway or in an urban scenario the observed channel conditions are a lot worse. This is not of direct influence to the validation, but lowers



(a) Sender First



(b) Sender Last, Interference Clear



(c) Sender Last, Interference Garbled

Figure 6.5: *Comparison of the Frame Capture results for different scenarios between the experiments performed in [3] and the estimated results based on our implementation.*

its value - due to the nature of the typical vehicular network channel it seems unlikely that frames can be captured successfully with SNRs between 0 and 2 dB.

The last difficulty in all of this is that there are no known other experiments that we could use to verify our capture implementation, let alone experiments done with real 802.11p hardware. (802.11p hardware, it must be noted, is still quite scarce because of the novelty of the new standard). We are thus stuck with a capture experiment with a lot of variables that are different from the simulation environment, but no suitable alternative we can use to validate our implementation.

6.3.3 Expected results

Due to these limitations it is impossible to validate our capture model with hard data - we just have to assume that the capture logic is at least reasonably accurate based on the results from the experiments. The capture results that we expect with our model are compared in Figure 6.5.

Please note that because of all the constraints mentioned in this chapter our implementation of frame capture is merely an approximation. The lack of available literature on this topic has made it a very challenging task to accurately model Frame Capture specifically for 802.11p. Our approximation should be further verified, preferably by experiments with real 802.11p hardware.

6.3.4 Results

In spite of all the shortcomings, we have performed simulations that mimic the experiment mentioned above, to show that our implementation behaves like we think that it behaves. We have created a network with three nodes, one receiver and two senders on either side. If the other sender was silent, the receiver could correctly receive a frame from either node. We let both nodes generate new beacons using a Poisson process, then waiting a quarter of a second (to make sure there would not be three frames associated with one capture event but always only 2) and then generating a new beacon again using a Poisson process. We logged all capture events and classified them accordingly.

In Figure 6.6 the capture results are shown. Because of the implemented capture thresholds at 8 and 16 dB, the Frame Reception Ratio jumps to 1 at these thresholds in the SLC and SLG cases; i.e. at the SNR where the capture thresholds are exceeded, the bit error calculation formulas do not have much influence on the results. This is also indicative of the shortcomings of the current BER implementation. In a real scenario these changes are not likely to be that abrupt, although for lower bitrates it is possible. As already mentioned, capture experiments with real 802.11p hardware should be performed to be able to fine-tune the model reliably.

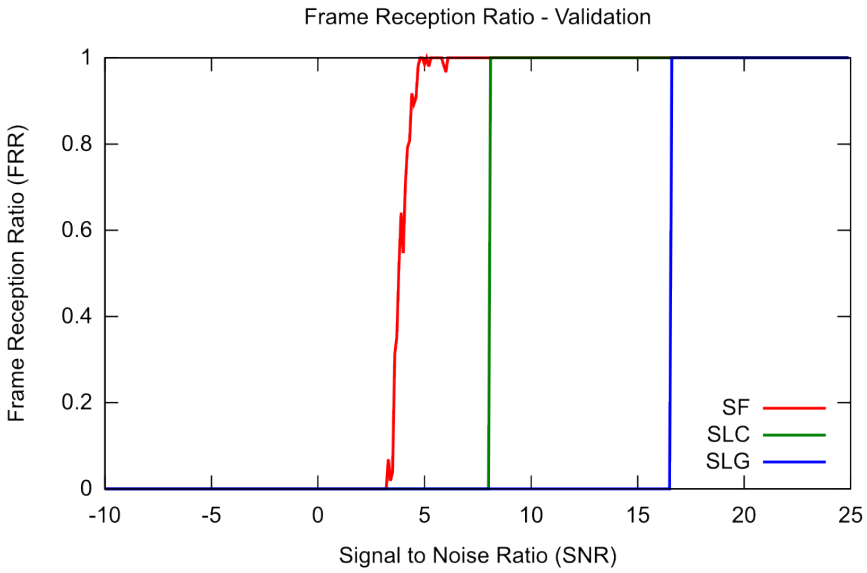


Figure 6.6: *Frame Capture simulation results of our validation experiments (all traffic at 3 Mbps). Because of the implemented capture thresholds the reception ratio jumps from 0 to 1 at these thresholds. This will not be the case at the higher bitrates, where the required SNR for correct reception is higher than the capture thresholds, not lower.*

7

Simulations

This part of the thesis covers the research questions regarding vehicular networks: how does Frame Capture influence IEEE 802.11p network performance? We suspect that in high-density traffic scenarios, Frame Capture has a significant effect on throughput in a vehicular network if the chipset supports it. If a frame is captured, a collision no longer necessarily means loss of bandwidth. Frame capture always occurs during a collision (i.e. a chipset could then capture a frame in favor of an other, weaker frame), so any scenario where many collisions occur is interesting for us. A vehicular network is such an environment, especially when there are many cars and they all broadcast beacons at a rate that is so high that the medium becomes saturated. By doing these simulations we demonstrate that Frame Capture has a measurable impact on performance in broadcast-oriented vehicular networks, and we show that the expected performance boost can vary among different chipsets.

We have made two significant changes to the simulator. First, we implemented Frame Capture so that not all frames during a collision are necessarily lost; second, we changed the bit- & packet error rate calculation formulas so that they better reflect 802.11p. Both these improvements should be considered separately when measuring the influence of either one; when introducing both changes simultaneously it is very hard to see which changes in the results reflect which changes in the implementation.

7.1 Simulation plan

For our simulation scenarios, the simulation parameters that are considered relevant are shown in Table 7.1. Regarding Frame Capture, we compare three different implementations, one with capture behavior that mimics Atheros chipsets, one that mimics Prism chipsets and one that has no capture behavior at all, i.e. all collisions result in the loss of both frames. We also study the impact of the new bit error approximation formulas to make sure they do not affect capture behavior in the simulator.

Some parameters that are important in a vehicular network have been fixed, i.e. not varied with different simulation runs. We did this for various reasons; mostly because changing these variables would not change the observed capture behavior or because the number of simulations that needed to be performed would grow too large. Some of the parameters listed could however have a great impact on the throughput and saturation rate of the medium and should be investigated further in the future.

As can be seen, we simulate that all cars are moving on a line (a highway with 1 or 4 lanes). We abstract away the real movement however by assuming that all nodes are moving in the same direction and at a constant speed, which causes the distances between the nodes not to change.

The following sections describe some of the important metrics, and how they will influence the simulation results.

7.1.1 Road vehicle density

The number of nodes (together with the traffic generation rate) is an important metric because they have great influence on the number of collisions that are likely to occur in a vehicular network. There are a few different scenarios.

If the distance between nodes (or node density) is approximately equal to the transmission range, there is only one hidden terminal for every sender-receiver pair. This is shown in Figure 7.1, where the blue node (the Sender node) tries to broadcast a frame. The two green nodes (the Receiver nodes) will receive it because they are in the communication range, the two red nodes (the Quiet nodes) will hear the transmission and can sense that the medium is busy. However the black nodes will not hear the transmission, and can start sending at any moment, affecting the reception of one of the green nodes. In this scenario (where there is one node in the communication range and one in the transmission range, there is exactly one hidden terminal for every ongoing transmission.

If we double the node density, the number of hidden terminals also doubles (if the distance between sender and receiver remains the same), as is shown in Figure 7.2. This shows that we can easily increase the effects of the hidden terminal problem by gradually increasing the node density. The number of hidden terminals in this case also depends

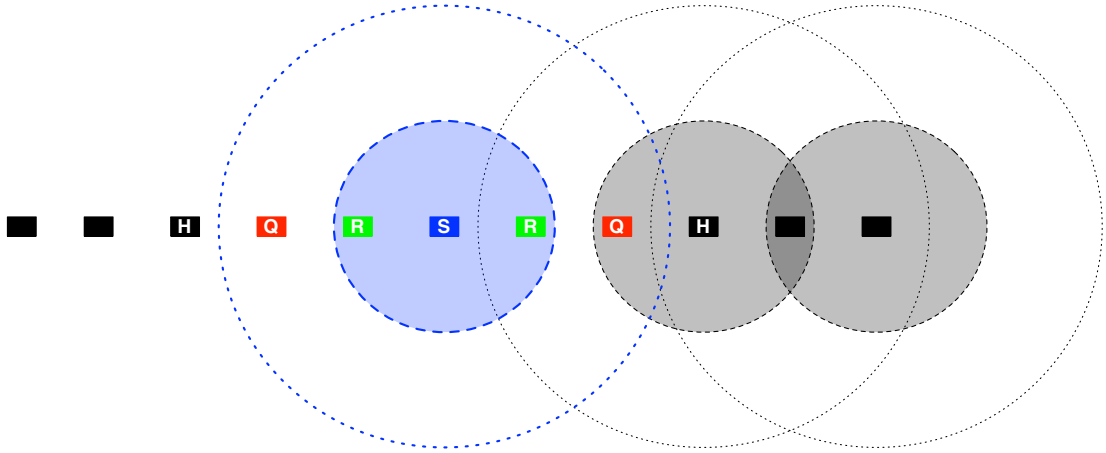


Figure 7.1: *The relationship between the node density and the hidden terminal problem for a low-density scenario. The blue node (S) transmits a message. The green nodes (R) can receive the message, the red nodes hear the message and will not transmit (Q), the black nodes (H) cannot hear the message and can thus at any moment start interfering with the ongoing transmission (i.e. they are hidden terminals).*

on which receiver is being considered; a node further away from the sender has more hidden terminals.

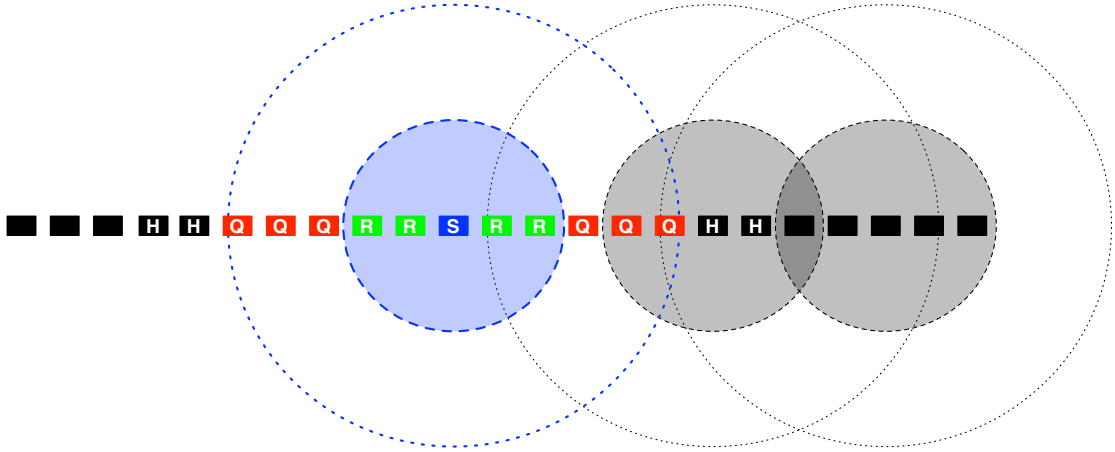


Figure 7.2: *The relationship between high node density and the hidden terminal problem. (S = Sender, R = Receiver, Q = Quiet, H = Hidden)*

Please note that these figures are a simplification of reality; under real circumstances there is a zone where the correct reception probability decreases gradually instead of a hard boundary as shown in the figures. Also, the relative size of the nodes to their communication range is not to scale.

7.1.2 Traffic generation rate

The traffic generation rate is not that important to us, because in the scenarios described above where movement between the nodes is abstracted away, an increased traffic generation rate would only create more collisions and capture events of exactly the same kind. If the scenario in the first image would use a traffic generation rate that is twice as high as the second image, the number of collisions caused by hidden terminals would be more or less the same assuming a Poisson process generated the beacons. However, the number of collisions because of simultaneously ending backoff counters could be different; since the contention windows do not scale linearly with either the node density or the number of collisions the nodes experience. Also, in our scenario where we have only broadcast traffic, the contention window does not scale at all, because broadcast traffic never gives feedback to the MAC layer (in the form of an ACK frame or the lack of it) as an indication that the transmission was lost. To simplify our simulation environment we assume a fixed traffic generation rate of 10 Hz. The relative impact of collisions between coordinated nodes vs. collisions caused by the Hidden Terminal Problem is discussed in Chapter 7.2.2.

7.1.3 Node chain length

Another important issue is the node chain length, i.e. the number of cars in total on the highway. We simulate a highway but we wrap around the simulation space in the x dimension, locating the nodes on a ring instead of on a straight line. This causes all the nodes to have the same number of neighbors and all transmissions to have the same number of hidden terminals and thus, the same collision probability. If we were to simulate on a straight line, the conditions at the start and the end of the line would be different from the conditions in the middle.

7.2 Basic parameters

The basic parameters of the simulation are described in this section. All nodes have fixed positions on a line and for every node the distance to their nearest neighbor is (in their lane) always equal. For the single-lane scenario we simulate a stretch of highway of 2000 m, and vary the number of nodes between 1 and 15 nodes per 100 meters. The multi-lane scenario is a stretch of 4 lanes, every lane 1000 meters long, and the node density varies here between 0.5 and 15 nodes per 100 meters per lane. This means that the total network has between 20 and 300 nodes in the single-lane scenario, and between 20 and 600 nodes in the multi-lane scenario.

All nodes broadcast 100 beacons of 3200 bits ($\approx$ 450 bytes with headers) at a rate of 10 Hz. The traffic is generated with a Poisson process, meaning that the times between

subsequent transmissions are exponentially distributed with $\mu = 0,1$. This distribution is used because it also reflects the arrivals of beacons in reality.

The thermal noise is set at -100 dBm. The capture thresholds are implemented as described in Chapter 6 and set at 0, 8 and 16 dB respectively. Radio switching times have been omitted, i.e. a receiving radio can switch between sending and receiving mode instantly. The receiver sensitivity is set at -94 dBm which is a typical value for 802.11 chipsets at their lowest bitrate (note that the receiver sensitivity is usually bitrate-dependent). The maximum transmission power of all nodes is 1000 mW, which is equal to 30 dBm, the maximum transmission power under the European regulatory domain [7]. As a comparison: the maximum vehicle transmit power in the United States is 28.8 dBm, equal to 750 mW ($10^{28.8/10} = 758$ mW) [58].

We use a path loss factor (α) of 3.2, which is typical for a relatively lossy urban environment [38]. While this path loss factor is too high for a typical highway with LOS communication (e.g. if two vehicles have their antennas mounted clearly above the vehicle and the antennas radiate in the right direction), we cannot assume that all transmissions will always have an LOS component. Vehicular networks will operate not only on highways but also in urban environments, or there could be a big truck positioned between the sender and receiver for example. This path loss factor means that the maximum interference distance of a node is 239 meters, as can be seen in Equation 7.1 [8, 51].

$$
\begin{aligned}
D_i &= \left(\frac{\lambda^2 \cdot P_t}{16\pi^2 P_{\min}}\right)^{1/\alpha} \\
&\text{where} \\
D_i &= \text{maximum interference distance,} \\
\lambda = \text{wavelength} &= \frac{c}{f^2} = \text{speed of light / frequency}^2, \\
P_t &= \text{transmission power (mW)} \\
P_{\min} &= \text{minimum receive power} = 10^{-R_{xsens}/10} \\
R_{\text{xsens}} &= \text{receiver sensitivity (dBm)} \\
\left(\frac{(2.998 \cdot 10^8/5.9 \cdot 10^9)^2) \cdot 1000}{16 \cdot \pi^2 \cdot 10^{-9.4}}\right)^{1/3.2} &= 239,46m
\end{aligned}
$$

This is a reasonable distance to work with, certainly in the scope of vehicular networks. With these parameters, we perform six single-lane simulations that can be seen in Table 7.2. Initially we performed each simulation 10 times. Based on these 10 repetitions we calculated the 95% and 99% confidence intervals, and they appeared to be very wide (see also 7.4). With this knowledge we finally performed every simulation run 5 times.

7.2.1 Multi-lane

After doing the single-lane simulation runs, the results indicated that we did not stretch the network to the limit from a capacity point of view. In order to put an even bigger load on the network and find out if Frame Capture is even more important if nodes do not just have neighbors in front of and behind them, but also on their sides, we decided to also perform simulations for a 4-lane highway. Because the simulation time increases exponentially with a linearly increasing node density, we have reduced the length of the highway to 1000 meters for these multi-lane scenarios. The node density per 100 meters of highway remains the same, so at the highest node density we simulate with 600 nodes. The distances between the nodes in one lane are still equal for every simulation run, but between lanes there are slight differences; the nodes in every lane are shifted a few meters (one-third of the distance between cars on one single lane) compared to the previous lane. In the multi-lane scenario we only look at the different behavior regarding Frame Capture, the bit error calculations can be sufficiently analyzed with just the single-lane scenario.

7.2.2 Hidden Terminal Problem

As we have concluded in Chapters 3 and 4, Frame Capture is closely related to the hidden terminal problem. There are two types of collisions that will occur in the network: the ones because of the hidden terminal problem (or HTP collisions), and the ones between coordinated nodes whose backoff counters end simultaneously (or Coordinated Node collisions). We suspect that Frame Capture is more important with HTP collisions than with CN collisions, because the observed SNR difference is larger with hidden terminals simply because the distance between the colliding nodes is larger. As can be seen from the capture thresholds, a chipset needs a certain SNR to capture a new frame. What we thus wanted to find out was how many collisions and captures originate from the hidden terminal problem, and how many originate from the coordinated nodes. To answer that question, we performed simulations where we 'turned the hidden terminal problem off'.

If many collisions occur because of hidden terminals, we suspect that Frame Capture can play an important role in improving the network's throughput. We can effectively disable the hidden terminals by lowering the receiver's sensitivity threshold, making a node able to hear frames at a larger distance. Note: if the simulation space is flat, and node A wants to broadcast, he must hear the ongoing transmissions of any other node of at least 2 transmission ranges away to disable the hidden terminal problem. This is because for any of those transmissions A might be a hidden terminal.

We can achieve this disabling of the hidden terminal problem using the *ConnectionManager* of OMNeT++. By lowering the *signal attenuation threshold* we can connect the nodes that are normally outside of the node's transmission range. All parameters that were mentioned before are still the same, but we added one more sensitivity threshold,

the HTP sensitivity. If a frame's power level is above this threshold, the receiver still marks the medium as busy, even though it is unable to receive the frame and outside of the transmission range of the sending node. We lowered this threshold to -105 dBm. If we input this new signal attenuation threshold in Equation 7.1, we see that the *ConnectionManager* will connect all nodes up to around 528,42 meters away. By setting the HTP sensitivity threshold to -105 dBm as well, we silenced all hidden terminals up to that distance (which is around 2 transmission ranges).

The only collisions that will still occur in this new scenario are the collisions between coordinated nodes. The difference in throughput measured and the collision probability of these new scenarios give us an indication of how many HTP collisions normally occur in the network. If the collision probability is a lot lower than in the normal scenarios, we know that the impact of the hidden terminal problem is normally very large.

Table 7.2 gives an overview of all the different simulations we performed, with the different capture behaviors and the different bit error rate calculation implementations. The 802.11b implementation is the current one, the 802.11p implementation is our modified implementation. The numbers in the table cells indicate the number of repetitions for each simulation. Note that the number of repetitions is not an important metric on itself, it is merely an indication and it is used to calculate the confidence intervals of the calculated metrics.

7.3 Metrics

During the simulation runs the collection of the right metrics is essential to give us insight in the performance of the network. OMNeT++ already has modules dedicated to metrics collection. From within any OMNeT++ module the function *recordScalar(name, value)* can be called, storing a name-value pair. These name-value pairs can be stored in plain text files, or in any other way that the user wants. We set up a MySQL database that recorded all the relevant scalars after every simulation run. Note that this occurred on a per-node basis, so aggregation of information was necessary after running the simulations.

Apart from the general network performance metrics, we also wanted to have more information on the capture events that took place. Therefore we also recorded a few metrics for every capture event that took place (e.g. whether the capture event was successful, what the observed SNR was, what the arrival time difference was etc.)

Both the capture metrics and the metrics we needed to provide us insight in the performance of the network are listed in Table 7.3.

Note that if multiple frames arrive during the reception of a frame, the mentioned arrival time difference and other frame state both apply to the *first* other frame that was involved in a capture event. For example if a strong frame B arrives during the

reception of a weaker frame A and the receiver captures frame B, and then during reception of B weaker frames C and D arrive at the receiver (in both cases the receiver stays locked to frame B). The number of capture events is 3 (A&B => B, B&C => B, B&D => B) and the capture count is 4 because there were 4 frames involved. However, the recorded arrival time difference and other frame state apply to A and B only. The SNR difference applies to all frames (A, C and D are summed as interference).

Not every single one of these metrics will be put to use directly and some metrics were only used for validation purposes, but the possibility exists that based on the results we want to zoom in on a certain part of Frame Capture; the broad metrics collection described here provides us the means to do so if necessary.

From these basic counters we can calculate the more advanced metrics which provide real insight in the network performance:

- *beacon success probability* - this is the percentage of the frames that are received correctly that also could have been received correctly. If a node broadcasts a beacon to 10 other nodes (all within communication range) and 9 receive it correctly, the beacon success probability is 90%. This percentage can be calculated by dividing, for every node, the number of successfully received frames by the number of frames that it *could* lock onto (i.e. frames arriving that were stronger than the receiver sensitivity). Since the throughput is hard to visualize in a broadcast network, this is our most important metric. The throughput in bytes/second doesn't say much in a network with only broadcast traffic, because for every transmission there is one sender and many receivers. This gives a distorted image of the capacity of the medium.
- *capture factor* - # successful captures / # frames received correctly - this tells us how many frames are received thanks to Frame Capture, i.e. how big the influence of Frame Capture is on the total reception process.
- *capture success probability* - # successful captures / # capture events - this tells us what the average probability of success is when a capture event takes place.
- *collision probability* - # collisions / # arriving frames (above sensitivity threshold) - the collision probability is a good indication of the load on the network and the probability of suffering from interfering frames during a transmission.

We can then visualize, for different node densities, what the impact of Frame Capture is on a vehicular network with the (aggregated) metrics described above.

7.4 Results

This section discusses the results of all the simulations that we performed. Note that in none of the graphs the confidence interval of the graphs is visualized. This is because the calculated confidence intervals (calculated with the t-distribution), even at 99%, were so small that they were not worth plotting. They were usually less than 0,2% for the various aggregated metrics.

7.4.1 Single Lane

In the single lane simulations, we compare the three different capture modes and the 802.11p and 802.11b bit error calculation implementations. In Figure 7.4 the most important metric is shown as a function of the node density: the beacon success probability (BSP). Remember that we defined the beacon success probability as the percentage of frames that have been received correctly, as compared to the total number of frames that *could* have been received correctly, i.e. that were strong enough to be decoded had there been no interference.

First, we show the collision probability for all single lane scenarios in Figure 7.3. Since all nodes generate traffic at the same rate and the node placement is the same for all scenarios, the collision probability is also the same. This gives us an indication of the load of the medium.

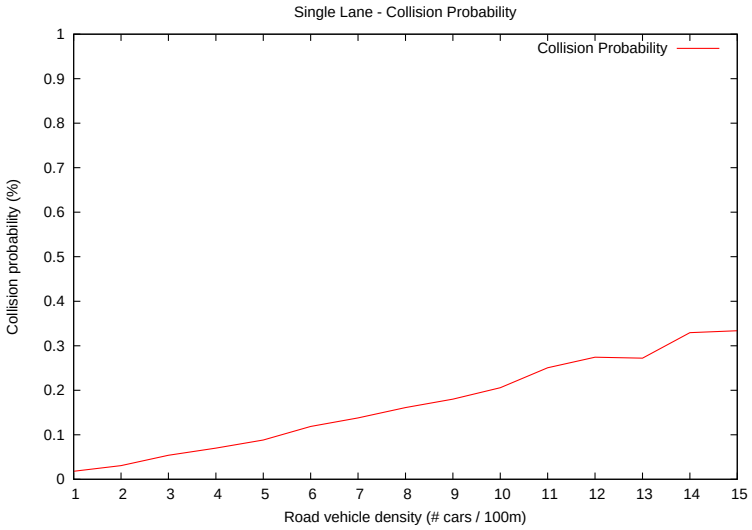


Figure 7.3: *Collision probability for single-lane scenarios*

The collision probability rises steadily when the node density increases, which is as expected because with increasing node density there are more hidden terminals. Also, since the contention window does not scale with the node density in a broadcast-only

network, the probability of collisions between coordinated nodes (i.e. because their backoff timers end simultaneously) also increases. It is not possible to see from this Figure how the collisions are distributed among these two collision types, we will try to answer that with the hidden terminal simulations.

If we then look at the BSP for the various scenarios shown in Figure 7.4, we see what we expected. In the figure there are three sets of lines; red, blue and green. Each color represents one of the frame capture varieties as displayed in the legend. It is logical that for all simulations the BSP decreases when the node density increases, but both implementations that are able to capture frames show a higher BSP, indicating the increase in performance. Note that Atheros performs better than Prism because of its ability to capture a stronger frame even while the receiver is locked to a weaker one. Compared to a chipset that performs no Frame Capture at all, both Atheros and Prism chipsets show a performance boost of at least 20%. The difference between Atheros and Prism are not spectacular but nonetheless noticeable; around 4% for the highest vehicle density. This BSP however has no one-on-one relationship to the degree of dissemination of information at higher layers (i.e. to what degree all nodes receive the information that is relevant to them). Depending on the application level protocol, a BSP of 50% can still be sufficient to meet the application requirements.

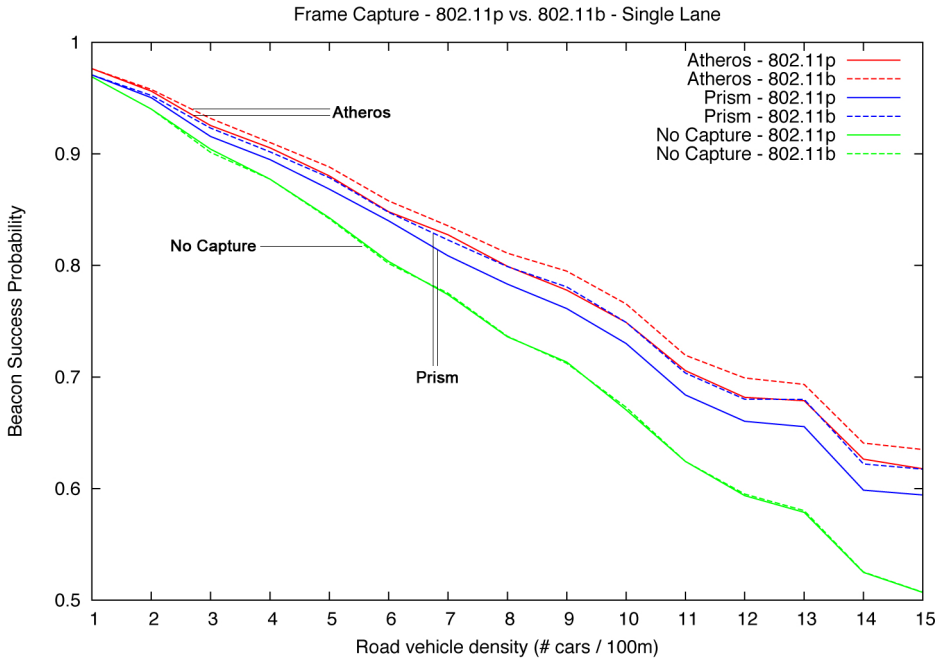


Figure 7.4: *Beacon success probability (BSP) comparison for the single lane scenarios between the three capture modes and both bit error calculation approximations.*

The dashed lines show the BSP when the 802.11b BER calculations are used, compared to the solid lines which show the BSP with 802.11p. It appears that 802.11b 'performs' a

bit better and this can be directly explained through Figure 5.5 - at SNR values approximately between 1 and 3 dB the 802.11b-specific formulas predict an almost negligible packet loss, whereas the new formulas still predict that a substantial part if not all of the frames are lost. Note that this is not really a performance gain because the 802.11b formulas simply do not apply to the physical layer in an 802.11p-based vehicular network. This merely shows that the implementation of the new BER and PER formulas does not affect the expected behavior regarding Frame Capture. The approximation for 802.11p is certainly closer to a real vehicular network than its 802.11b counterpart, but we suspect that when all channel effects are considered, the resulting approximation is even more pessimistic, and thus the BSP even lower.

If studied closely, it can be seen that for the implementation without Frame Capture the 802.11p and 802.11b BSP is almost the same for all node densities. The fact that this implementation shows no difference between 802.11p and 802.11b can be easily understood. From the literature in Chapter 4.2 and general literature on wireless communications we concluded that the preamble of a frame is the most important phase to ensure correct reception. The signal-to-noise ratio that a receiver needs to lock to a frame is above the range where the differences between 802.11p and 802.11b manifest, and this implementation discards all collided frames. We have double-checked this with our implementation, and all frames that are being processed by the receiver that did not suffer from interference (either by hidden terminals or by a simultaneously ending backoff counter) have a signal to noise ratio of around 6 dB. Around 6 dB packet error rates in both formulas from Figure 5.5 are negligible.

The difference between the two capture implementations (Atheros and Prism) is not very large - only a few percent. We can further visualize the impact of Frame Capture with Figure 7.5. This shows, as can also be more or less deduced from the BSP increase in Figure 7.4, that the influence of Frame Capture increases steadily with the increase in node density.

Another observation from this graph (and even more clearly in Figure 7.10 where we discuss the simulations without hidden terminals) is that the results are a bit 'wobbly'. This is probably caused by a slight methodical error in the approach w.r.t. the granularity of the node density we use and the deterministic node placement. Every node is placed at exactly the same distance to its neighbors. This causes some scenarios which have nodes at the borders of the transmission- and interference ranges to behave a bit different. We could mitigate this by placing the nodes not deterministically but using a sort of 'jittered' node placement, where we use a uniform distribution to shift the nodes a little bit to the right or to the left compared to their (current) deterministic placement. However, we would have needed to perform more simulations then to achieve the same confidence intervals.

To conclude; in these single-lane scenarios there is definitely a gain in BSP thanks to the capture behavior in some chipsets. The difference between Atheros and Prism chipsets is not spectacular, but still measurable, up to 8%. It will be interesting to see if the

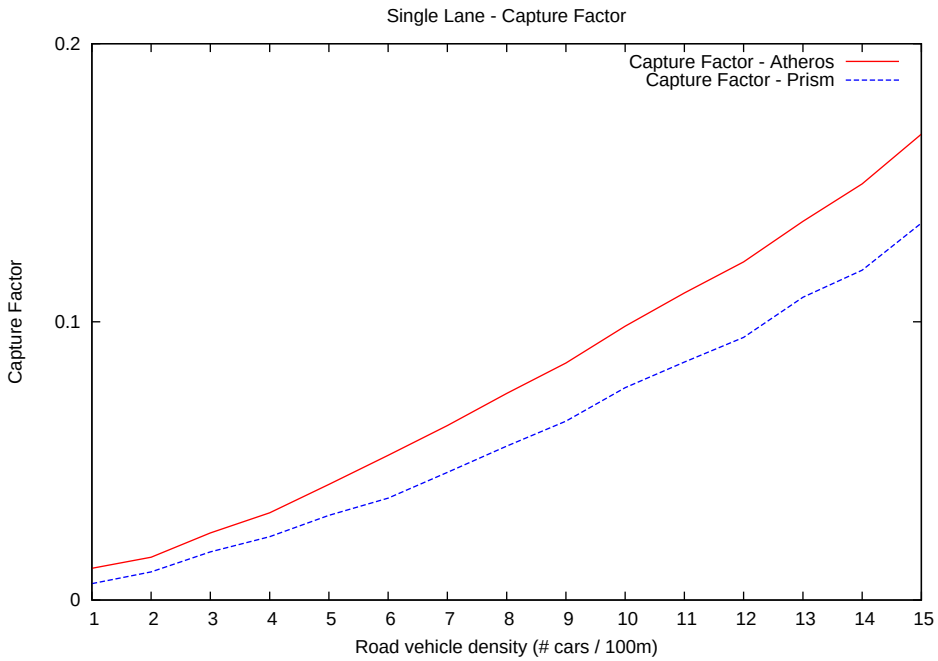


Figure 7.5: *The Capture Factor is the percentage of successfully received frames thanks to Frame Capture.*

impact of Frame Capture becomes even larger in a scenario with an even higher node density, such as our multi-lane scenario.

7.4.2 Multi Lane

In the multi-lane scenario, the number of hidden terminals is approximately 4 times higher than in the single-lane scenario. Linear increase in the number of hidden terminals means that the hidden terminal collision probability will also increase linearly, because hidden terminals are not synchronized to the transmitting node. The probability of collision between coordinated nodes will also increase linearly for a while, until the medium saturates. The medium is saturated if every node always has a packet waiting for transmission and all nodes are contending for the medium. In our simulation scenarios the contention window size never changes, because at the MAC layer there is no recovery or contention window adaptation because of 'lost' broadcast frames (a broadcast frame can lost at one node but received correctly at another). We visualize this with Figure 7.6; we have added the collision probability for the single-lane scenario for readability purposes. One would expect that when the contention window size doubles for every collision that is detected, the collision probability would not increase linearly in very dense networks. However because the contention window is fixed, it can be seen that in our case the collision probability does exactly that. For most node densities it is approximately 4 times higher and increases linearly until 10 nodes per 100 meter per lane. Beyond that point it flats out a little bit as the medium saturates. This flattening might be caused by dropping frames from the queue at the MAC layer, but we did not include a metric for that so with the simulation runs that we performed we cannot be sure that that is the case.

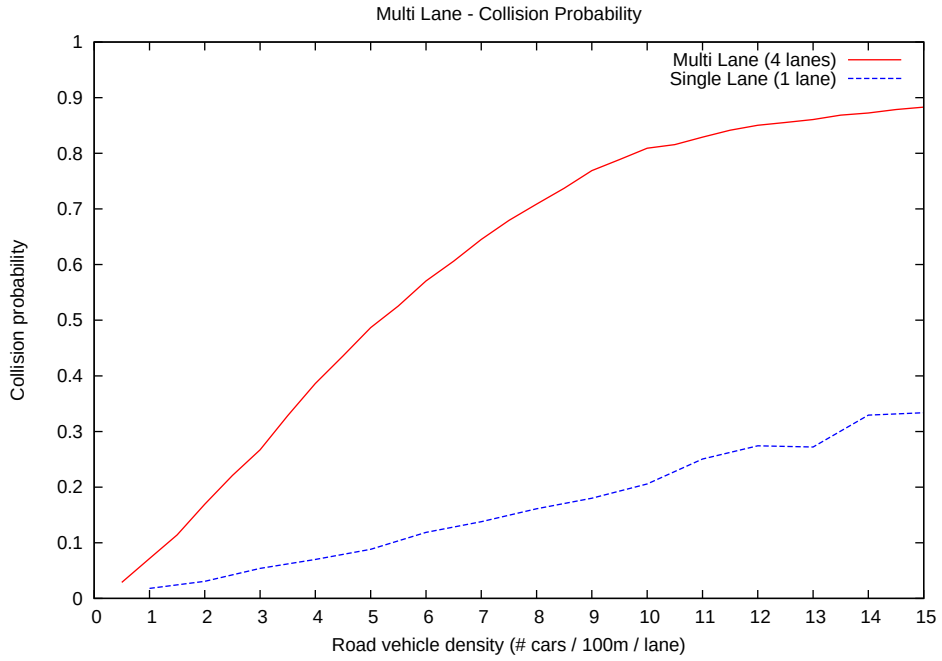


Figure 7.6: *Collision probability for single- and multi-lane (4 lanes) scenarios*

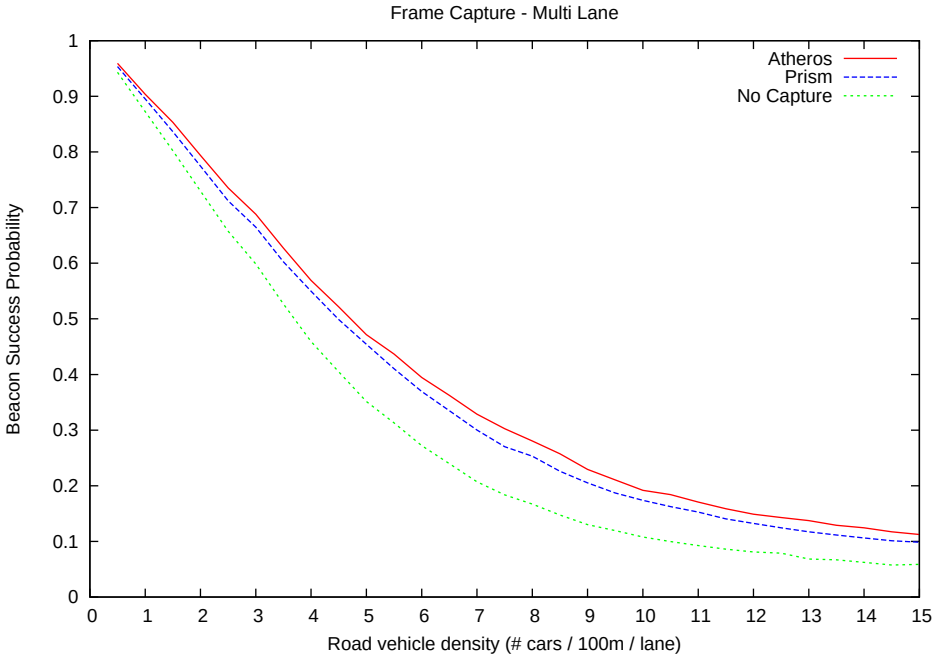


Figure 7.7: *Beacon success probability comparison for the multi lane scenarios, all with 802.11p bit error calculations.*

The BSP for the multi-lane scenarios is also very interesting to look at, it is shown in Figure 7.7. The figure shows that the BSP decreases rapidly with increasing node density, but also that Frame Capture plays an important role in mitigating that BSP decrease. There appears to be an optimum where the performance gain is the largest, around 7 nodes per 100m per lane. In all cases Atheros again slightly outperforms Prism. As the node density increases further it appears that the medium becomes saturated and the MAC layer starts to drop frames at the queue, causing the BSP to decrease less dramatically and more or less stabilize around 0.1.

The capture factor shown in Figure 7.8 shows the same image. Up to a node density of around 10 vehicles per 100m per lane the capture factor grows linearly with node density, but then the medium saturates and the capture factor stabilizes, or even appears to decrease slightly.

Obviously the increased number of neighbors every node has causes the number of collisions to increase rapidly and the BSP to decrease rapidly as well. This effect is amplified by the fact that all nodes have a fixed contention window. It would be interesting o compare the network performance with unicast networks (which use RTS/CTS and have fewer hidden terminals), and to vary the contention window size to see if Frame Capture remains equally important with a different contention window.

Since the transmissions from a node that is close by and a hidden terminal are likely

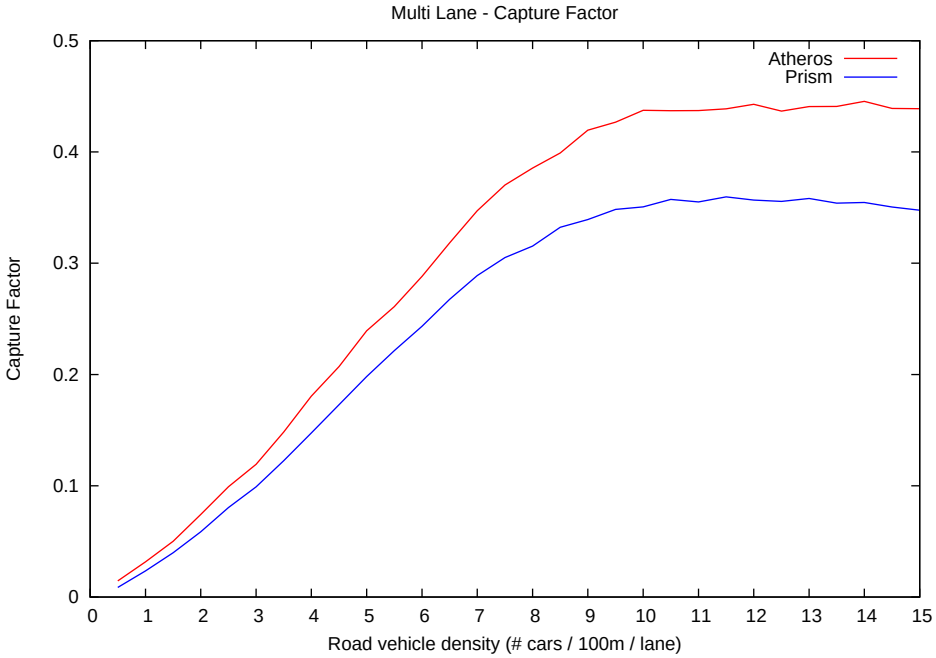


Figure 7.8: *Capture Factor for the multi-lane scenarios.*

to have significant difference in power levels, the probability that the strongest frame can be captured in a hidden terminal collision is higher than with collisions between coordinated nodes. And because the hidden terminal problem occurs less in unicast networks, there are probably fewer capture events in such a network. We can conclude though that for broadcast networks, Frame Capture definitely has a positive impact on performance.

7.4.3 Hidden Terminal Problem

In this section we describe the results from our simulations where we eliminated the Hidden Terminal Problem. We simulated both a single-lane scenario and a multi-lane scenario (with the Atheros chipset behavior and the 802.11p bit error approximations). The node placement, number of lanes and all other physical layer parameters are equal to the regular simulations, only the hidden terminals are silenced. The results are shown in the figures below.

The first graph (Figure 7.9) shows the collision probability for both the single- and the multilane scenarios. In both cases the collision probability is at least 50% smaller compared to the normal scenario (at the higher node densities). This also shows the relative impact of the hidden terminal problem - all collisions in the scenario without hidden terminals (the dashed lines) are caused by simultaneously ending backoff counters. Therefore the difference between the dashed and the colored lines of each color in Figure 7.9 demonstrates the relative contribution of the hidden terminal problem and the backoff counters to the total number of collisions. It can be seen that for both scenarios at least 50% of all collisions can be attributed to the hidden terminal problem. The numerical results (not shown) indicate that at 15 cars per 100 meters (per lane) the exact percentages of HTP collisions are 58,4% (single-lane) and 53,5% (multi-lane) respectively.

The reason that the percentage of HTP collisions in the multi-lane scenario is a bit lower is because of the combination of the fixed contention window size and the increased node density. Having more coordinated nodes with the same contention window size means that the probability of a coordinated node collision increases (that can be concluded from comparing the two dashed lines).

What is even more interesting is that this increase in node coordination actually decreases the beacon success probability for the multi-lane scenario. This can be seen in Figure 7.10. The single-lane scenario is not very surprising; because the medium is not saturated, the absence of hidden terminals increases the beacon success probability a little bit. However in the multi-lane scenario, where the medium does become saturated, the increased node coordination has actually negative effects. With our simulation, all nodes in the multi-lane scenario are coordinated, because the HTP sensitivity threshold causes nodes to indicate that their channel is busy up to 530 meters away, and the stretch of highway is 1000 meters. This has two effects:

- there is only 1 ongoing transmission in the medium at the same time; only if random backoff timers end simultaneously there are more than 1).
- nodes have to wait longer on average before they observe the medium as 'idle'. Because of this the buffer occupancy at the MAC layer is higher on average, meaning that more nodes have a beacon waiting for transmission. Unfortunately we had no instrumentation in place to measure this, so this could not be confirmed with hard evidence.

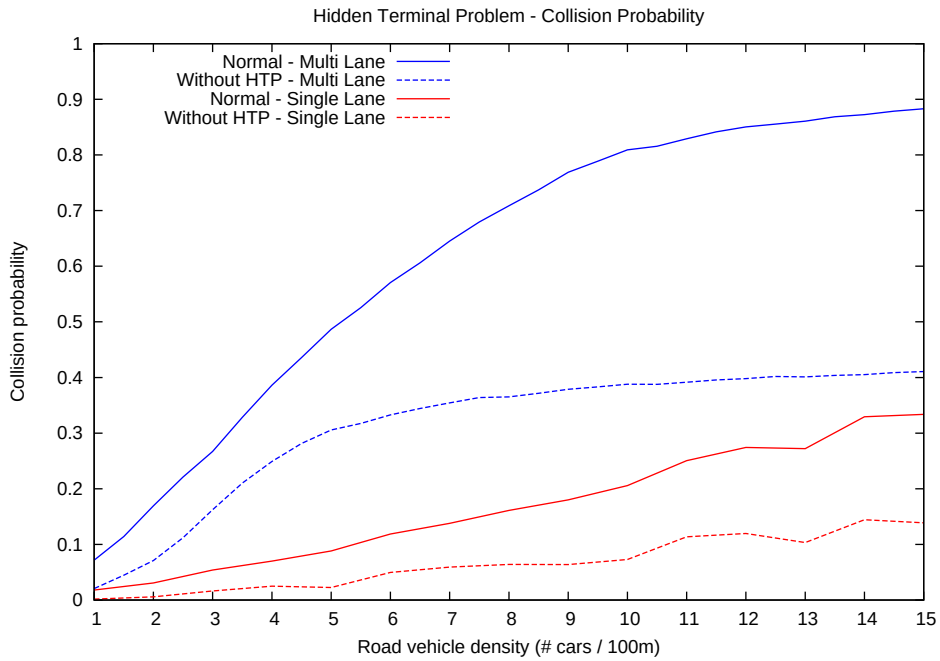


Figure 7.9: *Collision probability in the scenarios without hidden terminals. The upper two lines belong to the multi-lane scenario.*

Now if all nodes (including the hidden terminals) freeze their backoff timers when a transmission is ongoing somewhere in the network, and all nodes always have a transmission ready as soon as they sense the medium has become idle again, there are a lot of collisions between coordinated nodes. And these are exactly the type of collisions that are hard for Frame Capture to 'correct', because coordinated nodes usually have a more-or-less equal signal strength; the fact that they are coordinated indicates that the distance between them is smaller on average.

The changes to the scenarios mentioned above (silencing the hidden terminals and artificially making all nodes coordinated) have a strange effect: in our case, with a fixed contention window, the beacon success probability actually *decreases* instead of increases. Even though the probability of a collision is lower (the total number of collisions is of course also lower), a very large percentage of collisions that do occur cannot be mitigated by Frame Capture. This is also shown in Figure 7.11, where the Capture Factor in the multi-lane scenario without hidden terminal stalls very quickly and even decreases a little bit. This shows very clearly that Frame Capture is especially useful when resolving collisions caused by hidden terminals and that 'disabling' hidden terminals (and thus introducing node coordination where it is not necessary) will not solve the problem. How this manifests with different contention window sizes could be an interesting question for further study.

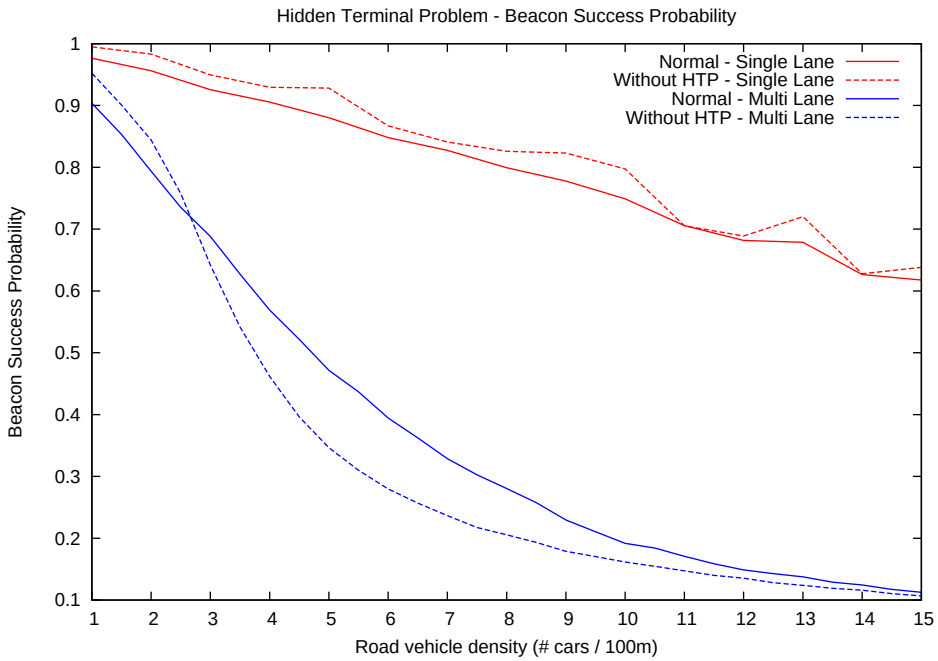


Figure 7.10: *Beacon success probability in the scenario without hidden terminal collisions. The upper two lines belong to the single-lane scenarios.*

The reason that we do not see this behavior in the single-lane scenario is that we do not enter the same 'collision regime' there. The single-lane highway is 2000 meters long instead of 1000 meters, meaning that not all nodes are coordinated. Basically what we see in every graph regarding the single lane metrics (the red lines) only represents the first part of the multi-lane scenarios, where the medium is not saturated yet and not all MAC queues are constantly occupied.

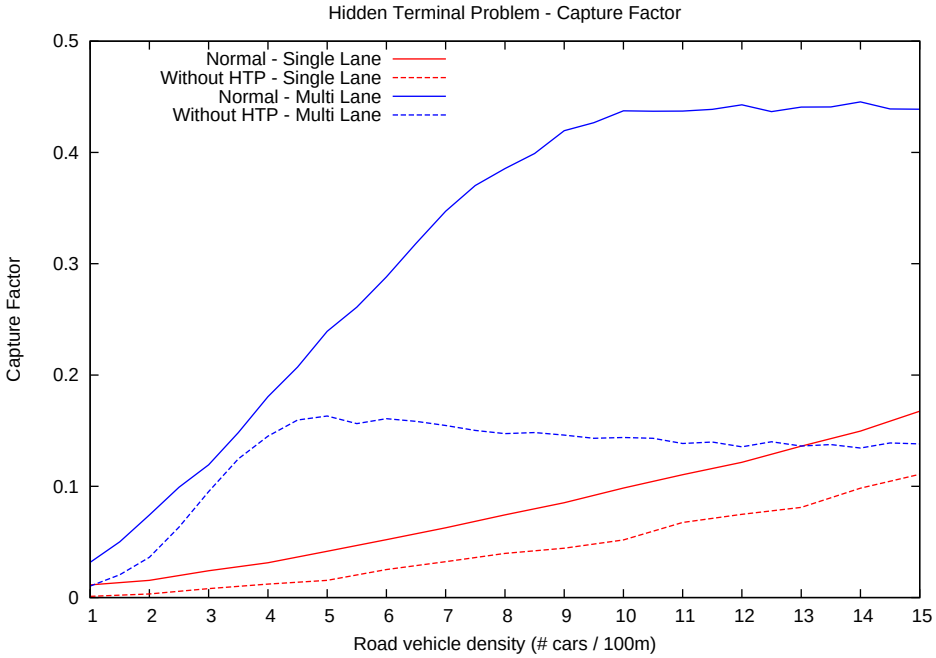


Figure 7.11: *Comparison of the Capture Factor in the normal scenario and without hidden terminal collisions. The upper two lines belong to the multi-lane scenarios.*

Table 7.1: *Frame Capture simulation scenarios*

Variable	Explanation
Frame Capture	We use three different Frame Capture implementations: One representing the Atheros behavior, one for Prism, and one which has no capture at all. This last implementation is not based on a real chipset but used as a reference to measure the performance boost caused by Frame Capture.
Bit Error Calculations	We use two implementations: the one based on 802.11b (current implementation), and our modified implemenation based on 802.11p.
Road vehicle density	We vary the road density between 1 and 15 cars per 100 meters in steps of 1 (single-lane) or 0.5 (multi-lane). One car per 100 meters is equal to very little traffic, and 15 cars per 100 meters is (roughly) equal to bumper-to-bumper.
Vehicle placement	This is not varied, the vehicle placement is deterministic. Nodes are placed at equal distances from each other.
Network traffic generation	Beacon generation is Poisson-based with fixed μ (=0,1). This is equal to 10 beacons per second on average.
Movement	We assume constant (and equal) speed for all vehicles.
Highway width	We use two different values here: we simulate a single-lane highway with only 1 lane, and a multi-lane highway with 4 lanes.
Beacon length	We use a fixed beacon length of 3200 bits
Data rate	The data rate we used is 3 Mbps in all cases.

Table 7.2: *Simulation scenarios (numbers indicate # repetitions)*

	Single-Lane		**Multi-Lane**	**w/o HTP**	
	802.11p	**802.11b**	**802.11p**	**SL 802.11p**	**ML 802.11p**
Atheros	5	5	3	5	3
Prism	5	5	3	-	-
No Capture	5	5	3	-	-

Table 7.3: *Simulation metrics*

Metric	Details	Scope
# frames sent		node
# frames received correctly	The number of frames that were received and were strong enough to be decoded correctly.	node
# frames received	This is the number of frames that the *ConnectionManager* delivered at the node's input gate (see also Chapter 5). This is not the number of correctly received frames!	node
# strong frames	The number of frames that arrived with a signal strength above the receiver sensitivity.	node
# frames locked	The number of frames that the receiver actually locked onto.	node
# collisions	The number of frames arriving when the receiver is already receiving an other frame.	node
# capture events	The number of times that the receiver captures a frame during a collision. This happens when, after the preamble of the colliding frame, the receiver is (still) locked on to a frame, either the new one or the old one. If the receiver was not locked to the old frame and the new frame is not strong enough (i.e. is not above the capture threshold), there is no capture event.	node
# successful captures	The number of times that a frame is captured and after that frame ends it is received correctly. Note that capturing a frame does not guarantee successful reception. Figure 6.6 seems to indicate that is the case, however that is only true for the lower bitrates. Also, if a lot of weak interference arrives after the preamble, the overall SNR will also be insufficient for correct reception.	node
arrival time difference	The difference in arrival time between two colliding frames involved in a capture event.	capture event
SNR difference	The signal to noise ratio of the frame that the receiver locks onto after the capture event.	capture event
capture count	The number of frames that arrived at the receiver during the entire reception of the captured frame.	capture event
capture result	Whether the captured frame was decoded correctly after its reception or not.	capture event
other frame state	What the state of the other (lost) frame was when the receiver captured the frame.	capture event

8

Conclusions

In this thesis, we investigated Frame Capture. All aspects of the phenomenon came to light, thanks to the available literature and the simulations we performed with our enhanced physical layer implementation.

As it appeared, there is no single Frame Capture implementation. The exact behavior is different among chipsets from various manufacturers [5, 15, 6, 3, 47]. We looked at the behavior in Atheros and Prism chipsets in particular, thanks to the available literature from the academic field on those two chipsets. Whether a receiver is able to capture a frame depends also on the signal-to-noise ratio, whether the frame receives interference during the preamble or not, and the difference in arrival times between the two colliding frames. However if a frame is captured, the literature indicates that this is always the strongest frame. Prism chipsets [59] can capture a stronger frame during a collision only if the stronger frame arrives first or during the preamble of the weaker frame. The behavior from Atheros chipsets [46] is slightly better from a performance point of view, because it is able to capture a frame independent of the state of the other frame. The exact workings of Frame Capture at a circuit level are hard to figure out; device & chipset manufacturers do not share their circuit designs unfortunately. This makes it harder to deduce the exact capture behavior, since it can only be observed through experiments, not by analyzing the design of the hardware itself.

We modeled the behavior of Frame Capture with so-called *capture thresholds*, which are SNR thresholds that the frame must exceed before the receiver is able to capture it. Depending on the state of the current frame, a different threshold is used to capture a new frame. If the receiver is still locked to the weaker frame, the new frame must be at least 8 dB stronger. However if the receiver is not locked to the weaker frame the

required SNR threshold is higher as well, because the noise becomes 'uncorrelated' and the receiver has more trouble suppressing it. In those cases we use a threshold of 16 dB. If there is no interference at all, the SNR must be at least 0 dB (i.e. stronger than the thermal noise).

We enhanced the current MiXiM physical layer implementation to enable frame capture using the thresholds described above. MiXiM previously (and by accident) only supported a part of the Prism-like capture behavior; it could only receive stronger frames if they arrived earlier, not during the preamble. During the performed simulations with Frame Capture it became clear that both Prism and Atheros chipsets perform at least 20% better than a chipset without any form of Frame Capture. Atheros performed an additional 5 to 8% better compared to Prism. We have also improved the bit-error calculation formulas that are used to determine or predict correct frame reception. The current implementation was based on IEEE 802.11b. The difference between both implementations is not spectacular, and the formulas are still not ideal because they assume Additive White Gaussian Noise as the only interference, but with their implementation we make a step forward towards better 802.11p channel modeling.

Apart from the normal simulations we also tested the importance of Frame Capture in combination with the hidden terminal problem. We did this by artificially disabling all nodes in the hidden terminal range. We increased the carrier sensing sensitivity so that nodes which would normally be hidden, would now be aware of the ongoing transmission and set their channel to busy, refraining from sending until the transmission was over. What we found there was really interesting; the remainder of collisions (that was caused by coordinated nodes) could usually not be resolved by Frame Capture, because coordinated nodes are usually closer to each other and their signal strengths differ less. This shows that in a broadcast network where nodes use a fixed contention window size, Frame Capture plays a vital role in resolving especially the hidden terminal collisions - the coordinated node collisions also occur, but the probability that Frame Capture can help in resolving those collisions is a lot smaller. This effectively shows the importance of Frame Capture in any scenario with lots of hidden terminals, whether it be a vehicular network or another type of network.

9

Future work

The impact and importance of Frame Capture in a vehicular network has been proven, but there are still many issues related to Frame Capture that could be studied further.

First of all, the number of experiments that we based our model on is a bit weak. More Frame Capture experiments, especially ones performed with 802.11p hardware, could help a lot in validating the model and further refining it. We have implemented Frame Capture now with hard thresholds that must be exceeded. Perhaps research that is oriented towards electrical engineering and signal theory could provide more insight in how a chipset uses the PLCP preamble to lock to a frame and under which circumstances. This knowledge could be used to refine our capture model.

It would also be interesting to investigate the impact of Frame Capture in other scenarios and with other physical layers, such as wireless sensor networks (where battery life is an important issue), and with vehicular networks that use mostly unicast traffic.

The channel model in the simulator could also be improved. Complicated channel effects such as multipath fading, scattering and narrowband interference are not yet incorporated in any way, mostly due to their complexity. There are however some models available, research to find out which one is most suited to a vehicular environment could be performed. Perhaps a good channel model could be the two-ray path loss model, which assumes an LOS component of the signal, and a reflection of the signal on the ground.

For Frame Capture in particular, the movement between nodes is not very relevant, that is why we chose to simulate without node movement. A good addition to the simulation

space could be to include not only movement in general, but also movement patterns from real drivers to the simulation. Part of this work is currently being done by our research group.

Other interesting directions in general for the physical layers of vehicular networks are not specific to 802.11p. Perhaps the usage of a physical layer with (Wideband) Code Division Multiple Access ((W)CDMA) is a very good alternative to OFDM - in WCDMA collisions are automatically resolved (because every node has its own code) and the system degrades gracefully [60]. Also, Multiple-Input, Multiple-Output (MIMO) could be investigated to increase channel capacity and scalability which might be needed for certain types of applications. Another application-dependent useful feature might be to investigate the usage of two directional antennas at the front and at the back of each vehicle, increasing the range of communication in one specific direction. There are many applications where a message (e.g. a warning message of an accident) should principally travel backwards, not forwards.

Bibliography

[1] Matthew S. Gast, *802.11 Wireless Networks*, O'Reilly, 2nd edition, 2008.

[2] Marc Torrent Moreno, *Inter-vehicle communications: achieving safety in a distributed wireless environment*, PhD thesis, Universität Karlsruhe, 2007.

[3] Jeongkeun Lee, Wonho Kim, Sung-Ju Lee, Daehyung Jo, Jiho Ryu, Taekyoung Kwon, and Yanghee Choi, "An experimental study on the capture effect in 802.11a networks", in *WinTech*, 2007.

[4] Ministerie van Verkeer en Waterstaat, Kennisinstituut voor Mobiliteitsbeleid, "Mobiliteitsbalans 2009", `http://www.verkeerenwaterstaat.nl/kennisplein/3/8/387913/Mobiliteitsbalans_2009.pdf`, June 2009.

[5] Jeongkeun Lee, Jiho Ryu, Sung-Jun Lee, and Taekyoung Kwon, "Simulating the 802.11a PHY model: how to make it accurate", *ACM*, 2008.

[6] Hoon Chang, Vishal Misra, and Dan Rubenstein, "A general model and analysis of physical layer capture in 802.11 networks", in *IEEE Infocom Proceedings*, 2006.

[7] Institute of Electrical and Electronics Engineers (IEEE), "IEEE Standard for Information Technology - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications - Amendment 7: Wireless Access in Vehicular Environments", July 2010.

[8] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P.T. Klein Haneveld, T.E.V. Parker, O.W. Visser, H.S. Lichte, and S. Valentin, "Simulating wireless and mobile networks in OMNeT++: The MiXiM vision", 2008.

[9] "OMNeT++: An extensible, modular, component-based C++ simulation library and framework", `http://www.omnetpp.org`.

[10] Karl Wessel, Michael Swigulski, Andreas Köpke, and Daniel Wilkomm, "MiXiM - The Physical Layer - An Architecture Overview", in *Proceeding of the 2nd International Workshop on OMNeT++*, March 2009.

[11] "IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical

Layer (PHY) Specifications", *IEEE Standard 802.11-2007 (Revision of IEEE Standard 802.11-1999)*, pp. C1 –1184, 12 2007.

[12] Daniel Minoli, *Hotspot Networks: Wi-Fi for Public Access Locations*, McGraw-Hill Professional, 2002.

[13] Gaddi Blumrosen, "The future of WiMAX", `http://www.thefutureofthings.com/articles/6361/the-future-of-wimax.html`, February 2009.

[14] Wayne Tomasi, *Electronic Communications Systems: fundamentals through advanced*, Prentice Hall, 5th edition, 2004.

[15] Jeongkeun Lee, Jiho Ryu, Sung-Ju Lee, and Taekyoung Kwon, "Improved modeling of IEEE 802.11a PHY through fine-grained measurements", *Computer Networks*, 2009.

[16] Vocal Technologies Ltd., "White paper - 802.11a wireless lan", `http://www.vocal.com/redirect/802_11a.html`, 2011 May.

[17] Charan Langton, "Coding and decoding with convolutional codes", `http://complextoreal.com/chapters/convo.pdf`.

[18] Christoph Sonntag, "Orthogonal frequency division multiplexing (OFDM) implementation as part of a software defined radio (SDR) environment", Master's thesis, University of Stellenbosch, 2005.

[19] Steve Halford and Karen Halford, "OFDM Architecture and Design Challenges", `http://www.commsdesign.com`, May 2002.

[20] Yunpeng Zang, Lothar Stibor, Georgios Orfanos, Shumin Guo, and Hans-Jürgen Reumerman, "An error model for inter-vehicle communications in highway scenarios at 5.9ghz", ACM, October 2005.

[21] H. S. Wilf, *Combinatorial Algorithms: An Update*, Society for Industrial and Applied Mathematics, 1989.

[22] Veronika Shivaldova, "Implementation of IEEE 802.11p physical layer model in SIMULINK", Master's thesis, Technischen Universität Wien, Fakultät für Elektrotechnik und Informationstechnik, June 2010.

[23] George C. Clark Jr. and J. Bibb Cain, *Error-Correction Coding for Digital Communications*, Plenum Press, 1981.

[24] John G. Proakis, *Digital Communications*, 4th edition, 1983.

[25] M. Pursley and D. Taipale, "Error probabilities for spread-spectrum packet radio with convolutional codes and viterbi decoding", *Communications, IEEE Transactions on*, vol. 35, no. 1, pp. 1 – 12, January 1987.

[26] Michele Weigle, "Introduction to vehicular networks", `http://www.csl.mtu.edu/cs5461/www/Slide/1-Intro-VANET.pdf`, 2007.

[27] Qing Xu, K. Hedrick, R. Sengupta, and J. Vanderwerf, "Effects of vehicle-vehicle/roadside-vehicle communication on adaptive cruise controlled highway systems", in *Proceedings from Vehicular Technology Conference*, 2002, pp. 1249–1253.

[28] Jan Rouwendal, "Driver behavior and congestion on highways", Ersa conference papers, European Regional Science Association, 1998.

[29] Atulya Mahajan, Niranjan Potnis, Kartik Gopalan, and An-I A. Wang, "Urban mobility models for VANETs", in *International Workshop on Next Generation Wireless Networks*, 2006.

[30] Marco Fiore, "Vehicular mobility models", in *Vehicular Networks*, Stephan Olariu and Michele C. Weigle, Eds., chapter 12. Chapman & Hall/CRC, 2009.

[31] David Murray, Michael Dixon, and Terry Koziniec, "Scanning delays in 802.11 networks", in *NGMAST '07: Proceedings of the The 2007 International Conference on Next Generation Mobile Applications, Services and Technologies*, Washington, DC, USA, 2007, pp. 255–260, IEEE Computer Society.

[32] Ian Tan, Wanbin Tang, Ken Laberteaux, and Ahmad Bahai, "Measurement and analysis of wireless channel impairments in DSRC vehicular communications", in *IEEE International Conference on Communications*, 2008.

[33] Patrick Robertson and Stefan Kaiser, "The effects of Doppler spreads in OFDM(A) mobile radio systems", 1999, vol. 1, pp. 329 –333 vol.1.

[34] European Conference of Postal and Telecommunications Administrations, "CEPT/ECC decision of on the harmonised use of the 5875-5925 MHz frequency band for Intelligent Transport Systems (ITS)", `http://www.erodocdb.dk/Docs/doc98/official/pdf/ECCDEC0801.PDF`, March 2008.

[35] "COMeSafety: Communications for eSafety", `http://www.comesafety.org`.

[36] Lothar Stibor, Yunpeng Zang, and Hans-Jürgen Reumerman, "Neighborhood evaluation of vehicular ad-hoc networks using 802.11p", in *13th European Wireless Conference*, 2007.

[37] Jijun Yin, Tamer ElBatt, Gavin Yeung, Bo Ryu, Stephen Habermas, Hariharan Krishnan, and Timothy Talty, "Performance evaluation of safety applications over dsrc vehicular ad hoc networks", in *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*. 2004, VANET '04, pp. 1–9, ACM.

[38] Theodore Rappaport, *Wireless Communications: Principles and Practice*, Prentice Hall, 2nd edition, 2001.

[39] Peter Sweeney, *Error control coding: from theory to practice*, Wiley and Sons, 2002.

[40] Jeongkeun Lee, Jiho Ryu, Sung-Jun Lee, and Taekyoung Kwon, "Revamping the IEEE 802.11a PHY simulation models", in *MSWiM '08*. 2008, ACM.

[41] Christopher Ware, John Judge, Joe Chicharo, and Eryk Dutkiewicz, "Unfairness and capture behavior in 802.11 ad-hoc networks.", in *IEEE International Conference on Communications*, 2000, vol. 1.

[42] Christopher Ware, Joe Chicharo, and Tadeusz Wysocki, "Modeling of capture behavior in IEEE 802.11 radio modems", in *IEEE International Conference on Telecommunications*, 2001.

[43] J. Boer, H. van Bokhorst, W.J. Diepstraten, A. Kamerman, R. Mud, H. van Driest, and R. J. Kopmeiners, "Wireless lan with enhanced capture provision", US Patent 5,987,033, September 1997.

[44] Naveen Santhapuri, Justin Manweiler, Souvik Sen, Romit Roy Choudhury, Srihari Nelakuduti, and Kamesh Munagala, "Message in message (MIM): A case for reordering transmissions in wireless networks", in *ACM SIGCOMM HotNets (Hot Topics in Networks)*, 2008.

[45] "Conexant", `http://www.conexant.com`.

[46] "Atheros Communications", `http://www.atheros.com`.

[47] Andrzej Kochut, Arunchandar Vasan, A. Udaya Shankar, and Ashok Agrawala, "Sniffing out the correct Physical Layer Capture model in 802.11b", in *ICNP: 12th International Conference on Network Protocols*, 2004.

[48] Mathieu Lacage and Thomas R. Henderson, "Yet another network simulator", in *Proceeding from the 2006 workshop on ns-2: the IP network simulator*, New York, USA, 2006, WNS2 '06, ACM.

[49] E. Martijn van Eenennaam, "Providing over-the-horizon awareness to driver support systems by means of multi-hop vehicle-to-vehicle communication", Master's thesis, Design And Analysis of Communication Systems, University of Twente, December 2008.

[50] "OMNeT++ User Manual", `http://omnetpp.org/doc/omnetpp41/manual/usman.html`.

[51] "MiXiM - A simulator for wireless and mobile networks using the OMNeT++ simulation engine", `http://mixim.sourcefourge.net`.

[52] "The network simulator - ns-2", `http://www.isi.edu/nsnam/ns/`.

[53] "The ns-3 network simulator", `http://www.nsnam.org/`.

[54] George S. Fishman, *Discrete-Event Simulation*, Springer, 1st edition, 2001.

[55] Andras Varga, "Using the OMNeT++ discrete event simulation system in education", *IEEE Transactions on Education*, vol. 42, pp. 11, 1999.

[56] "Open hardware abstraction layer (openhal)", `http://madwifi-project.org/wiki/About/OpenHAL`.

[57] Frank Brouwer, Irene de Bruin, João Carlos Silva, Nuno Souto, Francisco Cercas, and Américo Correia, "Usage of link-level performance indicators for HSDPA network-level simulations in E-UMTS", in *Spread Spectrum Techniques and Applications, IEEE Eighth International Symposium on*, 2004, pp. 844 – 848.

[58] Federal Communications Commission, "FCC Rule 90 Subpart M - Intelligent Transportation Systems Radio Service (Parts 90.371 to 90.383)", `http://www.gpo.gov/fdsys/pkg/CFR-2009-title47-vol5/pdf/CFR-2009-title47-vol5-part90.pdf`.

[59] "Intersil corporation", `http://www.intersil.com`.

[60] Jochen H. Schiller, *Mobile Communications*, Addison Wesley, 2nd edition, 2003.

[61] Krishna Sankar, "Minimum frequency spacing for having orthogonal sinusoidals", `http://www.dsplog.com/2007/12/31/minimum-frequency-spacing-for-having-orthogonal-sinusoidals`, December 2007.

Acronyms

ACC	Adaptive Cruise Control.
ACK	Acknowledgement.
ACR	Adjacent Channel Rejection.
AGC	Automatic Gain Control.
AP	Access Point.
AWGN	Additive White Gaussian Noise.
BER	Bit Error Rate.
BPSK	Binary Phase Shift Keying.
BSS	Basic Service Set.
CACC	Cooperative Adaptive Cruise Control.
CEPT	Conférence Européenne des administrations des Postes et des Télécommunications.
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance.
CTS	Clear To Send.
DCF	Distributed Coordination Function.
DIFS	Distributed Inter-Frame Space.
DS	Distribution System.
DSSS	Direct Sequence Spread Spectrum.
FC	Frame Capture.
FCC	Federal Communications Commission.
FEC	Forward Error Correction.
FFT	Fast Fourier Transform.
FRR	Frame Reception Ratio.
HPA	High-Power Amplifier.
HTP	hidden terminal problem.
ICI	Inter-Carrier Interference.

IEEE	Institute of Electrical and Electronics Engineers.
IFFT	Inverse Fast Fourier Transform.
ISI	Inter-Symbol Interference.
ITS	Intelligent Transport Systems.
IVC	Inter-Vehicle Communication.
LNA	Low-Noise Amplifier.
LOS	Line Of Sight.
MAC	Medium Access Control.
MANET	Mobile Ad-Hoc Network.
OFDM	Orthogonal Frequency Division Multiplexing.
PA	Power Amplifier.
PAPR	Peak-to-Average Power Ratio.
PER	Packet Error Rate.
PHY	Physical Layer.
PLC	Physical Layer Capture.
PLCP	Physical Layer Convergence Protocol.
PLL	Phase-Locked Loop.
QAM	Quadrature Amplitude Modulation.
QPSK	Quaternary Phase Shift Keying.
RSS	Received Signal Strength.
RTS	Request To Send.
SF	Sender First capture.
SIFS	Short Inter-Frame Space.
SLC	Sender Last, Interferer Clear.
SLG	Sender Last, Interferer Garbled.
SNIR	Signal to Interference and Noise Ratio.
SNR	Signal to Noise Ratio.
STA	Station.
UDP	User Datagram Protocol.
V2I	Vehicle to Infrastructure.
V2V	Vehicle to Vehicle.
VANET	Vehicular Ad-Hoc Network.

WAVE	Wireless Access in Vehicular Environments.
WBSS	WAVE Basic Service Set.
WLAN	Wireless Local Area Network.

A

Channel effects

An overview of the effects that a wireless propagation environment can have on an electromagnetic signal in the wireless medium.

- ***free space path loss*** - every signal experiences signal degradation because of the propagation through space. The received power P_r is proportional to $\frac{1}{d^2}$ where d is the distance between sender and receiver. This happens because the signal power spreads in all directions, so only a small fraction of the emitted power arrives at the receiver (like a small surface on a sphere) [60].
- ***shadowing*** - sometimes also referred to as shadow fading, is the attenuation caused by large objects (trees, trucks, walls). The object absorbs most of the signal energy. If the object is in the line of sight between sender and receiver this causes significant signal degradation.
- ***reflection*** - if an object is very large compared to the wavelength of the signal (such as very big buildings, the earth itself, or mountains, the signal is (apart from being partially absorbed) also reflected. This reflection is very useful if there is no line of sight between sender and receiver (such as between a laptop and the access point in an office environment).
- ***refraction*** - if the signal can propagate through an object, the object might cause refraction. This is basically what also happens when visible light passes through a prism or glass; the direction of the wave changes.
- ***scattering*** - if an object is a bit smaller (in the order of the wavelength or less), the wave is 'split up' or scattered into several weaker outgoing signals, in multiple

directions.

- ***diffraction*** - is similar to scattering, refers to the deflection of a signal at the edge of a large object, making the signal 'bend' around the edge of the object and continue in a scattered fashion.

Please note that the wavelength ($\lambda = \frac{c}{f}$, wavelength is speed of light divided by frequency) of a typical 802.11 signal is between 5 and 15 centimeters, depending on the 802.11 variation used.

These channel effects cause *fading* of the signal at the receiver. Fading is the concept of fluctuations (either in amplitude, phase or delay) of a radio signal over a short period of time. There are various types of fading:

- ***Slow vs. fast fading***. Slow fading is a type of fading where the amplitude and phase change are considered roughly constant during signal transmission. This can be because of shadowing, or if both sender and receiver are stationary, because of multipath fading (explained below). Fast fading occurs on a very small time-scale, causing deep fades for very short periods of time. Fast fading occurs a lot in dynamic (urban) environments with moving objects, such as cars, and/or moving senders and receivers.

- ***Flat vs. frequency-selective fading***. Flat fading occurs when all frequency components of the signal experience the same amount of fading. Frequency-selective fading is caused by *changes* (i.e. lengthening or shortening because of ionospheric movements) of the multipath components. The frequency at which the fade is deepest changes constantly

- ***Multipath induced fading*** is caused by interference between two or more versions of the transmitted signal. These multiple versions are created because the original signal wave is transmitted in all directions, and then reflected on all sorts of objects/surfaces, thus creating a multitude of waves all coming from the same source but all having traveled different paths.

These fading types can be modeled using different (statistical) models such as Rayleigh fading, Rician fading and Nakagami fading. Rayleigh fading is more suitable for an urban environment with lots of scattering, Rician is a better approximation if there is always a Line-Of-Sight component of the signal. Nakagami can be parameterized to approach both environments.

B

Orthogonality of multiple sinusoidals and their frequency spacing

For two sinusoidals to be orthogonal to each other, the following must be true:

$$\int_0^T cos(2\pi(f_1 t + \phi) \cdot cos(2\pi f_2 t) dt = 0 \tag{B.1}$$

where T is the symbol period, f_1 and f_2 are the frequencies of the two sinusoidals and ϕ is the phase difference of the first cosine. Integrating and applying the limits, this simplifies to

$$\begin{aligned} cos(\phi)\left[\frac{sin(2\pi(f_1+f_2)T)}{2\pi(f_1+f_2)} + \frac{sin(2\pi(f_1-f_2)T)}{2\pi(f_1-f_2)}\right] + \\ sin(\phi)\left[\frac{cos(2\pi(f_1+f_2)T)}{2\pi(f_1+f_2)} + \frac{cos(2\pi(f_1-f_2)T)}{2\pi(f_1-f_2)}\right] = 0 \end{aligned} \tag{B.2}$$

Also note that for any integer value of n, $sin(n\pi) = 0$ and $cos(2n\pi) = 1$. Then if we assume that $(f_1 + f_2)T$ is an integer, we see that a few terms in the equation above disappear because $sin(2\pi(f_1 + f_2)T) = 0$ and $cos(2\pi(f_1 + f_2)T) = 1$. Substituting simplifies Equation B.2 to

$$cos(\phi)\frac{sin\Big(2\pi(f_1-f_2)T\Big)}{2\pi(f_1-f_2)}+sin(\phi)\frac{cos\Big(2\pi(f_1-f_2)T\Big)-1}{2\pi(f_1-f_2)}=0 \tag{B.3}$$

So, for a random phase (ϕ) between 0 and 2π, the numerators of both fractions need to be 0 in order for the whole equation to be zero, and thus have orthogonality. So $sin(2\pi(f_1-f_2)T)$ has to be 0 and $cos(2\pi(f_1-f_1)T)$ has to be 1, this is the case when $2\pi(f_1-f_2)T=2n\pi$, n being an integer. This condition, $2\pi(f_1-f_2)T=2n\pi$, can be simplified to $f_1-f_2=\frac{n}{T}$. The minimum value of n is 1, so the minimum frequency difference at arbitrary phase difference between f_1 and f_2 is $\frac{1}{T}$, and the two sinusoidals are orthogonal at frequency differences $\frac{n}{T}$ [61].