# Hidden Terminal Detection in Wide-Area 802.11 Wireless Networks

A thesis

submitted in fulfilment

of the requirements for the degree

of

**Doctor of Philosophy**

in

**Computer Science**

at

**The University of Waikato**

by

**Scott McKenzie Raynel**

THE UNIVERSITY OF
WAIKATO
*Te Whare Wānanga o Waikato*

2012

## Abstract

The hidden terminal problem is an important issue in wireless networks based on the CSMA medium access control scheme. Hidden terminals pose a complex challenge to network operators trying to identify the underlying cause of performance issues.

This thesis describes new methods for the detection and measurement of the hidden terminal problem in wireless networks based on commodity hardware and software platforms. These new methods allow network operators to identify areas of a network where hidden terminals are likely to exist; detect instances of the hidden terminal problem occurring; and estimate the total impact hidden terminals are having on the performance of the network.

A new framework for measurement of wireless networks is described which provides a new approach to wireless measurement on Linux based wireless routers. The new framework is used to implement the methods and they are deployed across an operational commercial wireless network and are shown to be useful.

# Acknowledgements

This work would not have been possible without the help and support of a large number of extraordinary individuals. To those who have been there along the journey and provided support, encouragement and an understanding ear, thank you, I couldn't have done it without you.

Dr. Tony McGregor provided invaluable guidance, mentorship, feedback, support and motivation as my Chief Supervisor. My co-supervisors Dr. Matthew Luckie and Dr. Richard Nelson also provided feedback and support along the way. I would also like to especially thank Dr. Murray Pearson who first took me on as an undergraduate honours student and provided me with the initial guidance and inspiration to proceed into post-graduate study.

The members of the WAND Network Research Group in the Department of Computer Science at the University of Waikato, both support staff and fellow students, have been invaluable throughout the process. Thank you to all those involved, especially Jamie Curtis, Perry Lorier, Dean Armstrong, Sam Bartels, and many others.

At various stages I was financially supported by the Department of Computer Science, the University of Waikato and Rural Link Limited. I am sincerely grateful for their support.

My parents Bill and Margaret have always been a source of constant support, motivation and love and for that I am forever grateful. Finally, to my partner Emily, for the support, love and patience you've shown throughout, thank you.

# Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 The Problem

Hidden terminals are a classic problem [65] faced by wireless network protocols based on the Carrier Sense Multiple Access (CSMA) [29] medium access control scheme and have the potential to negatively impact the performance of such networks. CSMA-based wireless networks have become increasingly prevalent over the last 10 years with the rapid and wide-spread adoption of the IEEE 802.11 “Wi-Fi” [6] family of standards for wireless communications. Aside from its wide-spread use in home and office networks, IEEE 802.11 has proved to be useful in a multitude of other environments largely due to its low implementation costs and relative ease of deployment.

IEEE 802.11 has found particular momentum as a backbone and access mechanism for low cost rural and remote community networks and Wireless ISP (WISP) deployments, as can be seen in [10, 42, 62, 61] and many others. Such network deployments aim to provide internet connectivity into remote areas that do not have access to traditional broadband infrastructure. In many cases, remote areas with low population densities do not warrant the significant investment required to deploy traditional broadband infrastructure [39, 63, 37] and as such, IEEE 802.11 has proved fundamental in extending network reach into such areas through community involvement and partnership with local business and government.

These community networks are commonly built using commodity “off the shelf” hardware and free and open source software platforms. It is also common for such networks to be operated by groups of non-expert users. A lack of deep technical knowledge of underlying technologies can make diagnosis of network performance issues difficult, and hidden terminals in particular pose a complex challenge for non-expert users to diagnose as the root cause of performance issues.

The hidden terminal problem arises when a set of terminals in a CSMA network are unable to fully carrier sense each another. A CSMA network that includes hidden terminals is unable to operate at its maximum capacity as hidden terminals can cause simultaneous overlapping transmissions leading to collisions. Hidden terminals can be caused by a number of physical issues, from obstacles obstructing the signal path between terminals, to path attenuation over long distances.

Link distances in rural and remote networks are much greater than is usual for standard IEEE 802.11 deployments, both in the point to point backbone network as well as the point to multipoint access network. These long distances, coupled with high gain directional antenna required to achieve such distances, contribute to an increased likelihood of hidden terminals in these networks. As such, hidden terminals are an important and real issue which operators of rural and remote community networks need to be able to detect and measure.

Detection and measurement of the hidden terminal problem in community networks is an open problem and there currently exists no scalable, robust methods for detecting and measuring hidden terminals in this context. This thesis sets out to design, validate, implement and deploy methods for detecting the potential for and measuring the performance impact of hidden terminals in IEEE 802.11 wireless networks based on commodity hardware and software platforms, with a particular focus on enabling long term measurement on such networks. In particular it asks the question, "can hidden terminals be detected and can the performance impact of hidden terminals be measured in wireless networks using commodity hardware and software platforms?"

## 1.2 Overview of the Thesis

This chapter presents the thesis problem and the contributions this thesis makes to the field of hidden terminal detection in community wireless networks. The next chapter, *Background*, provides a discussion of wireless medium access techniques and provides an overview of existing literature on the effect of hidden terminals and the current state of hidden terminal detection and avoidance. In Chapter 3, *Detecting Hidden Terminals*, two new methods for the detection and measurement of hidden terminals are presented; the first is based on measuring network connectivity, and the second uses detailed packet timing analysis to detect hidden terminal transmissions and estimate the total collision rate due to hidden terminals. Chapter 4, *Validation*, describes a laboratory validation of the timing (second) method and shows that it is effective in detecting hidden terminals and estimating the total rate of collisions caused by them.

Chapter 5, *Implementation*, describes a new approach to performing wireless measurement on Linux based wireless routers. A new framework for implementing wireless measurement tasks is described and the two hidden terminal methods are implemented within it. The implementation of the methods focuses on ensuring the methods are able to be deployed on a real-world, operational network. The limitations of performing wireless measurement on a large scale are discussed and a comparison of existing wireless measurement techniques is given. The new framework and the implementation of the two new methods for hidden terminal detection are described.

Chapter 6, *Deployment Results*, presents results from the deployment of the methods across an operational wireless network and shows that the methods can be used to solve real-world network problems. Finally, Chapter 7, *Conclusions and Future Work*, concludes the thesis and presents possible areas of future work related to the outcomes of this thesis.

## 1.3 Contributions

This thesis investigates the detection and measurement of hidden terminals in community wireless networks based on commodity hardware and software. The core contributions of this thesis are:

1. A method for detecting the possibility of hidden terminals in a wireless network using network connectivity measurements;
2. A novel method for detecting the presence of hidden terminals using detailed packet timing analysis;
3. A novel method for estimating the impact of hidden terminals by estimating the total number of collisions being caused;
4. A laboratory validation of the hidden terminal detection and measurement methodologies;
5. A new Linux kernel measurement framework for instrumenting wireless network stacks;
6. An implementation of the hidden terminal detection and measurement methodologies within the kernel measurement framework; and
7. A real-world deployment of the methodologies on an operational wireless network.

The usefulness of the hidden terminal detection and measurement methods is shown in chapter 6 by the results gained from deployment of the methods on an operational IEEE 802.11 wireless network that serves mainly rural residential customers. The operators of the network were able to make changes to the network that improved the performance of the network based on new information gained from the deployment of the methods.

The general focus in this thesis is to develop, test, implement and deploy new methods for the detection and measurement of the hidden terminal problem in operational IEEE 802.11 wireless networks based on commodity hardware and software platforms. The focus on operational networks puts specific constraints on the implementation and hence a new approach to performing wireless measurement is required. A new framework for large scale wireless measurement is developed as part of this thesis to enable the implementation and deployment of the hidden terminal methods. The new measurement framework and the hidden terminal detection methods themselves have been presented in [54]. Additionally, data gathered by a deployment of the measurement framework has been used elsewhere to provide long term wireless link-level data to a study that challenged existing path-loss prediction models [45].

# Chapter 2
# Background

The hidden terminal problem[1] is an artefact of the Carrier Sense Multiple Access (CSMA) [29] medium access scheme used by modern wireless technologies. This thesis is an investigation into the detection and measurement of the hidden terminal problem in CSMA-based IEEE 802.11 wireless networks. This chapter aims to provide motivation for the need for a robust, reliable and scalable system for detecting and measuring the hidden terminal problem in IEEE 802.11 wireless networks.

This chapter presents a review of current literature on the topic of random access channel acquisition, the effect the hidden terminal problem has on IEEE 802.11 networks, its proposed solutions and current techniques for its measurement. First, the evolution of the IEEE 802.11 medium access scheme is presented through a short history of random-access channel acquisition methods. Second, the hidden terminal problem is more precisely described in the context of IEEE 802.11 wireless networks as well as the effect hidden terminals can have on IEEE 802.11 networks. Third, the IEEE 802.11 RTS/CTS protection mechanism is evaluated. Existing solutions to avoid hidden terminals at the transport, medium access and physical layers are then evaluated. Finally, existing techniques for measurement of the hidden terminal problem are discussed.

There have been many studies into the effect of hidden terminals and many solutions to the problem have been proposed. However, very few of the proposed

[1]The hidden terminal problem is commonly referred to as the hidden *node* problem, however this thesis uses the original terminology defined by Kleinrock and Tobagi.

solutions are feasible for implementation within existing IEEE 802.11 networks while limiting cost and maintaining compatibility with existing devices. These constraints limit the ability of network operators to avoid hidden terminals in their networks. Despite work towards a solution, the hidden terminal problem still exists in operational IEEE 802.11 networks and its detection and measurement is important.

## 2.1 A brief history of wireless medium access schemes

The study of random packet-based medium access schemes for wireless networks is relatively new. In 1969 at the University of Hawaii, researchers developed the ALOHA system, known now as pure ALOHA. Many improvements have been made since then however many of the same problems that were present in ALOHA are still present in modern medium access schemes.

The following sections outline the evolution of wireless medium access schemes leading up to the IEEE 802.11 Distributed Co-ordination Function (DCF) from its origins in the ALOHA system.

### 2.1.1 The ALOHA System

The ALOHA system [3] (now known as "pure" ALOHA) was the first attempt at providing true random packet based access to a shared wireless medium in a point-to-multipoint manner. Previous to ALOHA, radio systems either used frequency (FDMA) or time (TDMA) division techniques to share the medium. These schemes often result in poor channel utilisation in cases where only a small proportion of the terminals in the network have any data to transmit as they allocate the shared resource independent of each terminal's activity. Random access schemes allow for more efficient re-use of the radio resource as terminals are allocated the medium as required [66].

The ALOHA system does not attempt to avoid collisions. It instead relies on stations to re-transmit packets for which no positive acknowledgement is received. If an acknowledgement is not received within some time period, the

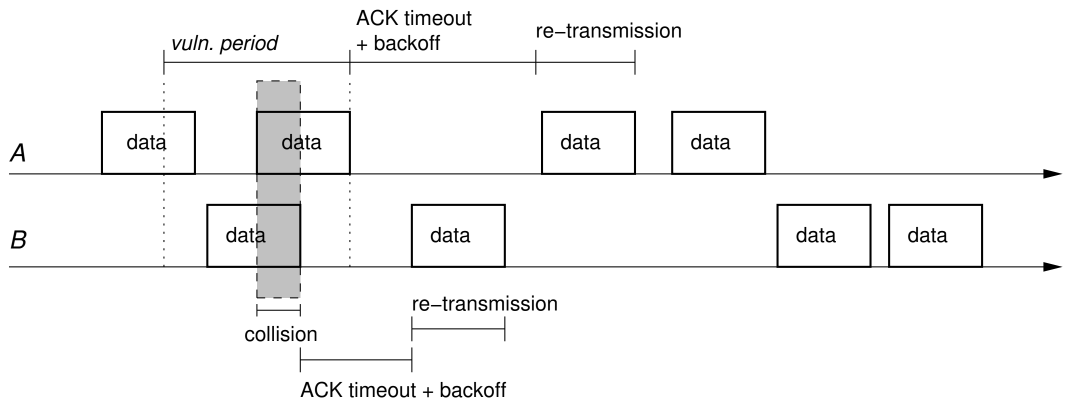

Figure 2.1: Two Pure ALOHA terminals $A$ and $B$ access the channel in a random and unsynchronised manner. This leads to a collision shown by the shaded area due to terminal $B$ beginning a transmission within the *vulnerable period* of $A$'s second data transmission. Each of the terminals begins an ACK timer after the end of their transmissions. No ACKs are received so each goes into independent random backoff states. Terminal $B$ has selected a shorter backoff than terminal $A$ so re-transmits first. The transmission is successfully ACKed. Terminal $A$ completes its backoff and retransmits its data frame.

station assumes that a collision has occurred and will wait a random amount of time before retransmitting. This random backoff is crucial to avoid repeated collisions. Given the unsynchronised nature of pure ALOHA, once a terminal is transmitting again any other terminal may also transmit causing yet another collision. Packets in the ALOHA system have a fixed length, and communication from and to the central point is performed on separate channels; therefore acknowledgement frames from the central point do not collide with data packets being sent by remote terminals. This behaviour is shown in Figure 2.1.

While ALOHA provided the first true random access scheme and enabled low cost resource sharing, the high probability of collision limited effective use of the transmission medium under moderate to heavy loads. It was shown in [29] that the maximum efficiency achievable by ALOHA was 0.184, where 1.0 indicates perfect scheduling and full use of the total available channel capacity.

In 1975, Roberts *et al.* [55] proposed Slotted ALOHA to improve on the efficiency of ALOHA. In Slotted ALOHA, the channel is split into discrete time slots. Terminals may only transmit at the start of a time slot and packet sizes are fixed to be one time slot in length. This avoids the problem of collisions occurring due to overlapping transmissions, however terminals may still collide at the beginning of the slot boundaries and may continue to collide

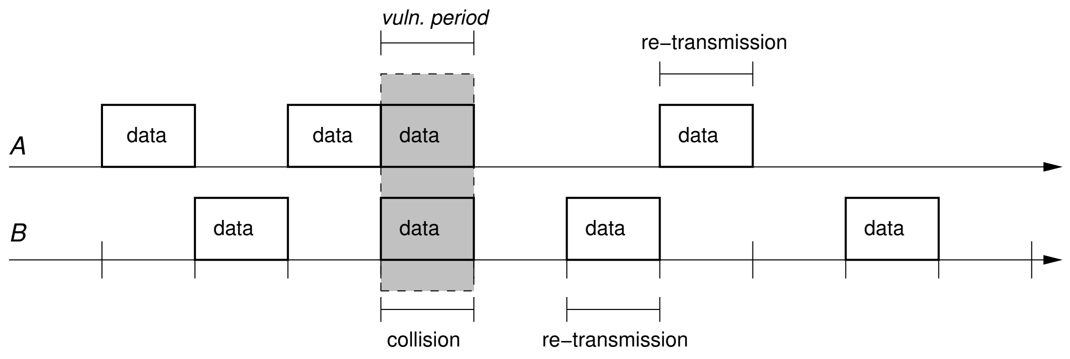

Figure 2.2: Two slotted ALOHA terminals are transmitting packets on slot boundaries. Their transmissions only overlap when they both transmit during the same slot. This reduces the *vulnerable period* to a single packet length and improves maximum achievable channel usage.

as they attempt to re-transmit.

Slotted ALOHA improves the channel efficiency over pure ALOHA by doubling ALOHA's maximum achievable channel usage to 0.368 [29]. This improvement is due to the reduced "vulnerable" period that collisions could occur in.

A later variant of ALOHA attempted to overcome the problem of repeated collisions by adding slot reservation to Slotted ALOHA. In Reservation ALOHA Borgonovo *et al.* [12] propose that when a collision occurs a Collision Queue Table (CQT) is constructed at each colliding station in which each of the retransmissions are scheduled using a priority assignment. This requires each station to know which other stations were involved in the collision. It is suggested that a small TDMA window is used to signal the existence of transmissions in each slot so that each station can know which other stations are attempting transmission and create the CQT accordingly.

The ALOHA system can be considered the first true random access packet switched medium access scheme for wireless networks. It sparked a new way of thinking about access to shared computing resources and was the catalyst for the explosion in wireless computer communication that was to come over the next several decades.

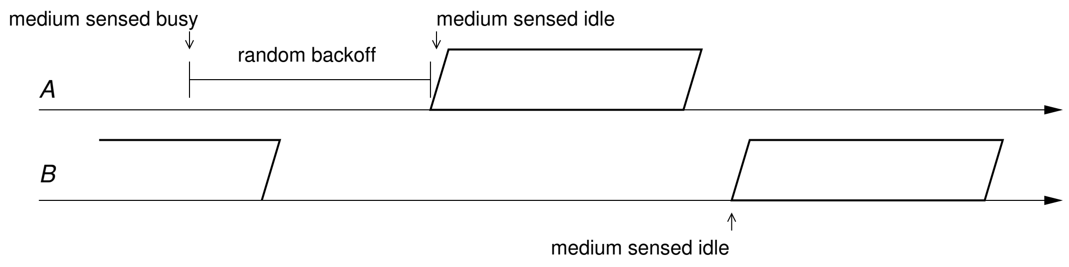

Figure 2.3: A simple example of non-persistent CSMA. Terminal $A$ has a packet ready to transmit and senses the medium. Terminal $B$ is already transmitting so terminal $A$ senses the medium as busy. It waits a random amount of time before sensing the medium again, at which point it is idle and so transmits its packet, avoiding the collision with terminal $B$.

### 2.1.2 Carrier Sense Multiple Access (CSMA)

In order to address the lack of synchronisation amongst participants in the ALOHA system, Kleinrock and Tobagi proposed a new access scheme, Carrier Sense Multiple Access (CSMA) in 1975 [29]. CSMA differs from ALOHA in that rather than sending randomly, a station with a packet of data to transmit first senses the medium to see if it is busy. If it is, then the station defers its transmission to allow the currently transmitting station to complete, thus avoiding a collision.

There are several operational modes of CSMA described by Kleinrock and Tobagi which dictate how the station responds to a busy channel [29]. In *non-persistent* CSMA, when the station senses the channel is idle it may transmit; however, if it is busy the station must reschedule another transmission attempt at some random time in the future. In *p-persistent* CSMA when the channel is sensed busy the terminal will continue to sense the channel until it becomes idle and then transmit with probability $p$. The focus of this thesis is on non-persistent CSMA access as used by the IEEE 802.11 Distributed Coordination Function (DCF).

By adding carrier sensing, CSMA allows participants in the network to synchronise their transmissions with each other and in perfect conditions avoid collisions, greatly improving the channel utilisation. However, as mentioned by Kleinrock and Tobagi, CSMA operates under the assumption that all stations are within carrier sense distance of one another. If they are not, then collisions may still occur. This is a manifestation of the hidden terminal problem and will be discussed at length later in the thesis.

Compared to ALOHA and Slotted ALOHA, CSMA achieves a much higher maximum channel capacity usage. It is shown in [29] that when the channel is able to be immediately sensed idle after a transmission (i.e. in the case that $a = 0$ where $a$ is propagation delay), non-persistent CSMA can achieve a theoretical channel usage of 1 for an offered channel load of infinity.

However, in a more realistic setting where propagation delay is taken into account, non-persistent CSMA achieves an effective channel usage of 0.815 when $a = 0.01$. Interestingly, for large values of $a$ such as in satellite networks, ALOHA and Slotted ALOHA become more efficient than CSMA. Given that the operation of ALOHA does not rely on propagation delay it is able to maintain a constant channel usage whereas CSMA degrades as $a$ increases.

For ground based radio networks where $a$ is relatively small, such as typical IEEE 802.11 installations, CSMA is superior to any of the ALOHA variants in terms of its effective channel use. However its reliance on carrier sensing can cause issues when stations are not within carrier sense distance of each other.

### 2.1.3 Medium Access with Collision Avoidance for Wireless

CSMA relies on transmitters being able to detect an ongoing transmission in order to avoid collisions. In 1994, Bharghavan *et al.* [11] argued that it is not always the case that a transmitter can detect such interference, for example, when two transmitters are hidden from one another. It should instead be the responsibility of the intended receiver to ensure that senders do not transmit simultaneously as the receiver has a more complete knowledge of its current channel state. Bharghavan *et al.* proposed the Medium Access with Collision Avoidance for Wireless (MACAW) protocol which allowed receivers the opportunity schedule transmissions and inform other potential senders of ongoing transmissions.

MACAW is a slotted, *non carrier sensing* protocol much like Slotted ALOHA. It differs from CSMA/CA in that it uses an RTS-CTS-DS-DATA-ACK sequence to transmit data. A station with data to transmit first sends a Request To Send (RTS) frame. The receiver sends back a Clear To Send (CTS) frame

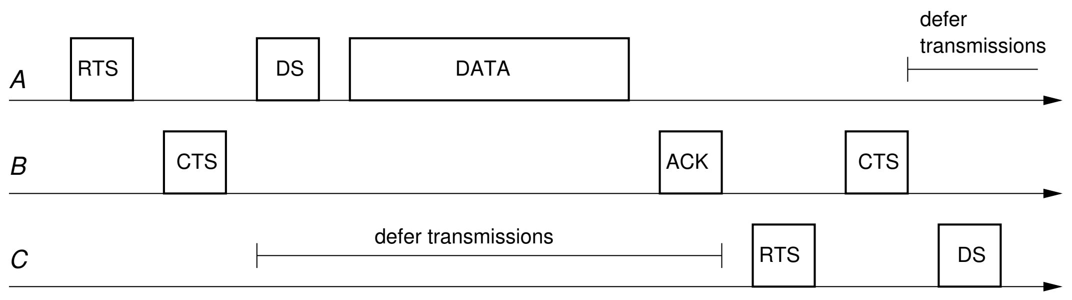

Figure 2.4: A simple example of MACAW. Terminal $A$ has a packet to transmit and so enters the RTS-CTS-DS-DATA-ACK sequence. When Terminal $C$ hears the CTS it defers any transmissions that it may have until the end of the ACK frame. In this way, the receiver, terminal $B$ is synchronising the transmissions of terminals $A$ and $C$.

which informs all other stations that they are to be quiet while the sending station transmits. The sending station then sends a Data Sending (DS) frame followed by the actual data fragment. Once the data fragment is received correctly the receiving station sends back an Acknowledgement (ACK) frame.

MACAW improves over the ALOHA variants by providing a mechanism for stations to reserve the channel during data transmission, hence decreasing the probability of collisions. It also improves over CSMA in the case where terminals are hidden from one another. However it does so at the expense of channel time given to the RTS/CTS frame exchange which ultimately leads to reduced channel efficiency in situations where stations are not hidden from one another.

### 2.1.4 IEEE 802.11 Distributed Coordination Function (DCF)

In 1997 the IEEE ratified 802.11 [6], the first of several specifications that standardise physical (PHY) and medium access (MAC) layer protocols for wireless local area networks. At the heart of IEEE 802.11 is the Distributed Coordination Function (DCF). The DCF is a combination of CSMA with collision avoidance, paired with an optional simplified version of MACAW.

IEEE 802.11 DCF uses CSMA as part of its channel access mechanism to avoid collisions in the first instance and provide fair access to the medium by employing a random delay before attempting to transmit, coupled with a random backoff in the event of a collision.

IEEE 802.11 DCF uses a simplified version of MACAW to improve performance in the presence of hidden terminals. When the medium is sensed idle with CSMA, a station may enter a RTS-CTS-DATA-ACK exchange. The RTS/CTS protection mechanism is used to propagate "virtual carrier sense" information to those stations that are not within carrier sense range of the transmitter. Each station maintains a Network Allocation Vector (NAV) which is updated whenever a packet is heard. If, for example, a station receives a CTS frame destined for another station which is hidden from it, it will examine the CTS *duration* field which contains the time the channel is scheduled to be busy and update its local NAV to account for the busy channel time that cannot be physically carrier sensed. This virtual carrier sensing mechanism is used along with CSMA/CA to attempt to avoid hidden terminal collisions. The usefulness of the RTS/CTS protection mechanism is discussed further in section 2.2.2.

## 2.2 The Hidden Terminal Problem

None of the access schemes that have been discussed are collision free. Even with carrier sensing, collisions may still occur. Two stations may both sense that the channel is idle at the same time and then transmit together, causing a collision. A random access delay reduces the probability of this occurring.

The more likely scenario is that a second terminal that is not within carrier sense distance of a transmitting terminal senses that the channel is idle when it is in fact not and proceeds to transmit, causing a collision at the receiver. This is the hidden terminal problem and has been the subject of much research into its effect on network performance as well as potential solutions to it.

The following sections give an overview of the existing literature on the effect of hidden terminals on CSMA and IEEE 802.11 networks, the usefulness of the existing RTS/CTS protection mechanism as well as a number of proposed solutions to the problem at different layers of the OSI networking stack.

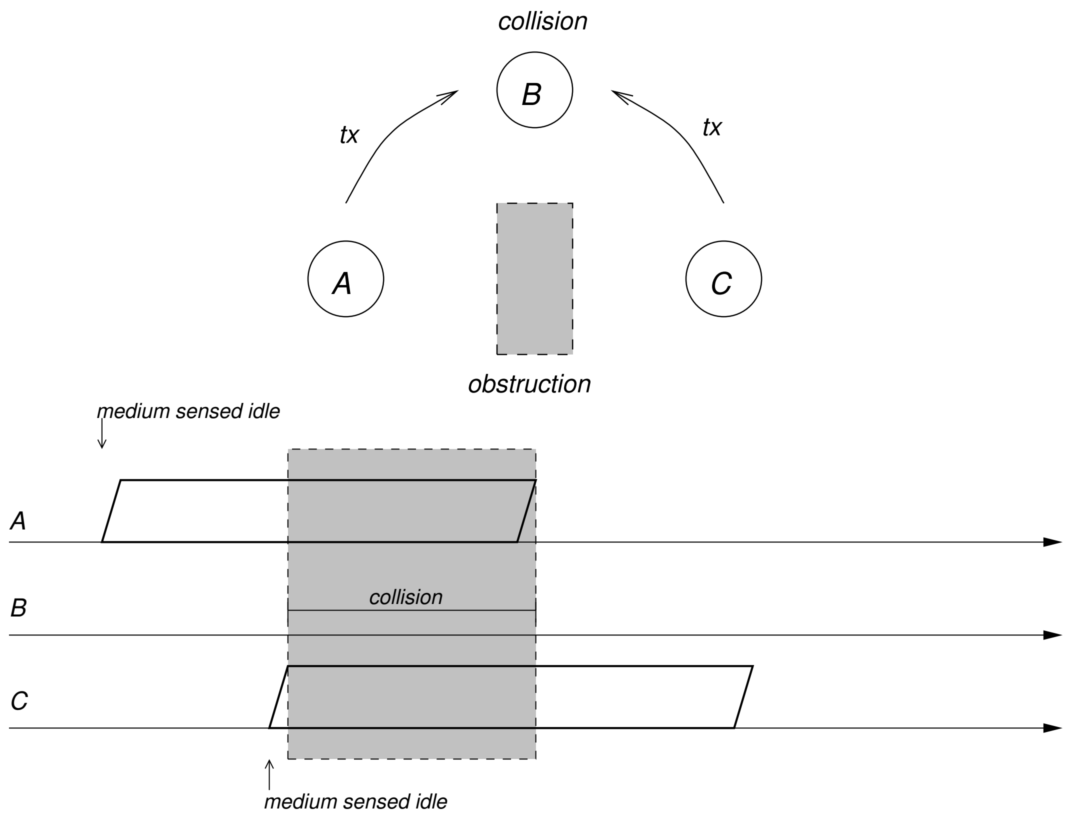

Figure 2.5: The hidden terminal problem in a CSMA network. Terminals $A$ and $C$ both have packets to transmit. Because they are *hidden* from one another, they both sense the medium as idle and transmit simultaneously, causing a collision at terminal $B$.

### 2.2.1 The Effect of Hidden Terminals

The effect of hidden terminals on packet switched random access networks is extensively documented in the literature. Tobagi and Kleinrock were the first to show that hidden terminals had an effect on their CSMA scheme and noted that in the presence of hidden terminals, CSMA degrades to pure ALOHA in the worst case [65]. This section outlines a number of the studies that have been carried out on the effect hidden terminals have on wireless networks. The negative effect hidden terminals can have on a wireless network highlights the need for tools to discover and measure the impact of hidden terminals on the performance of operational networks.

In one of the earliest studies of the effect of hidden terminals on IEEE 802.11 networks, Khurana *et al.* [28] showed in 1998 that when approximately 30% of stations in the network are hidden from one another throughput drops to as low as 22% of available capacity. The authors proposed that the throughput decrease is “acceptable” [28] when up to 10% of stations are hidden from each other.

Moh *et al.* [40] provided an early comparison of the HIPERLAN [1] and IEEE 802.11 MAC protocols under hidden terminal conditions using simulation. Their work focussed mainly on HIPERLAN and the effect hidden terminals have on real-time communication support. However they showed that IEEE 802.11 suffers severely under hidden terminals in terms of throughput. They also note that increasing frame size does not reduce the effect of hidden terminals.

Ray *et al.* [51] used queuing theory and simulation to analyse the effect of hidden terminals on packet collision probability, delay and maximum throughput. Their work shows, among other results, that in a linear topology hidden terminals can cause IEEE 802.11 nodes to saturate at load as low as 15% of their capacity.

Hung *et al.* [24] developed a Markov model to describe the performance of IEEE 802.11 in the presence of hidden terminals. The authors use the model to estimate the optimal stable transmission rate for each source in a network. They propose that this method could be used to implement rate control algorithms that deal with hidden terminals more effectively.

Kosek *et al.* [31] studied the effect hidden terminals have on star topologies when using 802.11e for QoS. The authors used an ns2 [41] simulation to show that, when using 802.11e priority queues, high priority flows achieve a lower throughput and higher loss rate in the presence of hidden terminals than when priority queuing is not used.

### 2.2.2 The Usefulness of the DCF RTS/CTS Mode

The IEEE 802.11 Distributed Coordination Function (DCF) uses the CSMA/CA medium access scheme, which leaves it vulnerable to hidden terminal performance degradation. In order to address this, the IEEE 802.11 DCF provides an RTS/CTS mechanism based on MACAW to attempt to propagate virtual carrier sense information to all stations in the network. However, it has been shown that the DCF RTS/CTS mechanism does not completely protect against hidden terminals. This section gives an overview of some of the literature surrounding the effectiveness of the RTS/CTS protection mechanism in IEEE 802.11 and shows that the RTS/CTS mechanism is not an effective tool to guard against hidden terminals.

Xu *et al.* [71] show that the IEEE 802.11 RTS/CTS mechanism is not effective at preventing hidden terminal collisions. They show that the RTS/CTS mechanism fails in cases where a transmitter is out of receive range of the intended receiver and hence does not receive the CTS frame. This causes the transmitter to sense the channel as idle and transmit, potentially causing a collision.

Ray *et al.* [50] showed that RTS/CTS doesn't protect hidden terminals in ad-hoc networks when a terminal fails to correctly receive RTS/CTS exchanges because of other ongoing transmissions in the area. Their work shows that the RTS/CTS mechanism is itself prone to hidden terminal collisions.

Shih *et al.* [60] also show that RTS collisions can still occur as the initial RTS frame is only protected by physical carrier sensing. The RTS and CTS protection frames are generally sent at a low bitrate in order to maximise receive Signal to Noise Ratio (SNR). This however results in a longer time on the channel and hence a longer period in which the initial RTS frame is vulnerable to collision.

Ware *et al.* [68] studied the effect of the channel capture phenomenon on the usefulness of RTS/CTS exchanges. The authors found that in the presence of hidden terminals whose SNRs differ, terminals with stronger SNRs are able to effectively capture the channel even when the RTS/CTS handshake is in use. This work was extended in [69] to show that hidden terminals with only small differences in SNR can render an RTS/CTS mechanism useless, effectively blocking the channel to weaker users.

Ray *et al.* [49] investigated the overhead involved in the RTS/CTS mechanism. They showed that, in ad-hoc networks, hidden terminals that cause RTS collisions can in some cases lead to a deadlock whereby no stations are able to transmit for long periods of time.

Pananastasiou *et al.* [44] looked at the effect the RTS/CTS handshake has on TCP "goodput", or application level throughput. They found that in many situations the use of RTS/CTS can decrease TCP goodput significantly. The authors also noted the presence of RTS collisions in multihop scenarios, leading to a decrease in effective throughput.

This section has shown that the RTS/CTS protection mechanism is not effective at combating the negative effects of hidden terminals. As such, measurement of the hidden terminal problem remains important. The following three sections discuss proposals that have been made to either solve the hidden terminal problem or reduce the negative effect it has on the network. They can be broadly categorised into transport layer, medium access control layer and physical layer solutions and will be discussed as such.

### 2.2.3 Transport Layer Solutions

Many researchers have studied the effect of hidden terminals - and more generally the effect of interference and collisions - on TCP. TCP assumes a model of a link that IEEE 802.11 links do not follow. Specifically, loss on an IEEE 802.11 link does not necessarily indicate link congestion and could instead be caused by hidden terminal collisions, among other explanations such as random bit errors, multi-path fading and external interference. Solutions to the hidden terminal problem proposed at the transport layer have focussed heavily on controlling TCP's congestion window in order to provide fairer access to the medium to all stations when hidden terminals are present.

Xu *et al.* [72] show that hidden terminals cause serious TCP instability and unfairness in multi-hop environments due to the large number of in-flight TCP segments causing RTS collisions and note that limiting the TCP congestion window dampens this effect.

Fu *et al.* [16] further investigates limiting TCP's maximum congestion window size to allow for better spatial channel reuse in the presence of hidden terminals in chain topologies. They find that for a given wireless network topology there exists a window size at which TCP achieves a maximum throughput and that increasing the window size from there leads to link-layer contention and a degradation of TCP service quality. It is noted that when the window size is unbounded, TCP usually increases the window size far beyond the optimal value, causing a degradation in service. As a solution to this, Fu *et al.* suggest methods for TCP to co-operate with the IEEE 802.11 MAC directly. Link Random Early Detection (RED) [16] allows the MAC to mark packets in much the same way as traditional RED [15], however the probability of marking the frame is based on the average number of retransmissions at the link-layer. This allows Explicit Congestion Notification (ECN) [48] enabled TCP flows to adjust their sending rate without losing packets.

Papanastasiou *et al.* [43] propose an alteration to TCP's congestion avoidance phase to slow the rate of congestion window increase. Their Slow Congestion Avoidance (SCA) mechanism limits the rate at which the congestion window is increased. An SCA sender maintains an SCA parameter such that the congestion window is increased every SCA + 1 RTT. An SCA value of 0 is equivalent to the default operation of TCP Reno. While the authors show that slowing the growth of the congestion window does indeed lead to more efficient spatial reuse and hence greater TCP goodput, they do not propose a method by which to set the SCA parameter. However, SCA improves on Link RED [16] in that it does not limit the maximum TCP congestion window size.

While these transport layer solutions provide some protection against the negative effect of hidden terminals they do so under a number of assumptions. First, they assume that all traffic that is competing for the medium is doing so using the same transport protocol, namely TCP. They also require modifications to the TCP implementation on the end hosts and require the TCP implementation to have an explicit knowledge about the underlying link. Third, because TCPs flow control is end-to-end, they require the sending host to be aware of the presence of wireless hops.

While cross layer communication is possible, it breaks the standard layering model which is adhered to by most network stacks. However, researchers have proposed that the only way to achieve fairness in mobile ad-hoc wireless networks is to move to a stack which is cross-layer aware and that the traditional strict layered model simply cannot cope with the complexities of the wireless medium [19]. In the area of wireless sensor networks where terminal deployments are chaotic and unplanned, leading to significant hidden terminal problems it is generally accepted that cross-layer aware network stacks are the only way to achieve reasonable performance [25].

It is likely that wireless links are not the only type of links between the sending and receiving end hosts. It is often impractical to expect end hosts to be aware of the presence of wireless hops along the path and modify their TCP algorithm accordingly.

### 2.2.4 MAC Layer Solutions

The hidden terminal problem is an artefact of random access medium access schemes. As such, there have been many proposed solutions to the hidden terminal problem in the form of MAC layer solutions. One such solution is the RTS/CTS scheme based on MACAW which is discussed earlier in this chapter.

One of the first proposals to avoid the hidden terminal problem was made by Tobagi and Kleinrock [65]. They identified the hidden terminal problem in their original paper on CSMA and proposed a solution they called Busy Tone Multiple Access (BTMA) [65]. BTMA uses a separate channel as a busy tone channel on which a terminal transmits a busy tone during reception of a packet. During this time all other transmitters who are within carrier sense distance of the receiver are informed that a transmission is taking place (which they may not have otherwise been aware of using plain CSMA if they were outside of carrier sense range from the transmitter). This improves synchronisation amongst senders and avoids collisions, improving channel efficiency. The authors show that BTMA with hidden terminals performs almost as well as CSMA without hidden terminals.

The most significant cost to implementing BTMA is the additional busy tone channel. This requires multiple radios per terminal, one for reception on the data channel and another for the concurrent transmission on the busy tone channel. This extra cost and complexity makes BTMA infeasible to implement as a low cost solution to the hidden terminal problem, especially on networks that already use single radio CSMA technology. A similar receiver-based busy tone approach was taken by Wu *et al.* [70].

The Floor Acquisition Multiple Access (FAMA) scheme as described by Fullmer *et al.* [17, 18] is a receiver initiated single channel BTMA mechanism. The previously described MACAW protocol is a FAMA variant. FAMA combines non-persistent carrier sensing with the RTS/CTS exchanges of MACAW and is a predecessor of IEEE 802.11 DCF.

A number of token-ring style replacements to the DCF have been suggested, usually in connection with multi-hop mobile ad-hoc networks in order to solve quality of service issues with contention based systems, [35, 58].

An interesting approach was recently suggested by Al-Mefleh and Chang [4] in which non-hidden terminals help each other retransmit faster when collisions occur due to hidden terminals. A terminal that successfully receives a frame that was not acknowledged will re-transmit the frame immediately after the next frame it has in its transmit queue. This allows the non-colliding terminal to take control of the channel sooner than the colliding terminal as it does not have to wait the extra backoff time that occurs after a frame fails to be acknowledged. The scheme remains compatible with existing DCF implementations and the co-operative retransmission technique results in improved throughput, delay and fairness.

Finally, adjustment of the physical Carrier Sense Threshold (CST) can lead to performance improvements in networks with hidden terminals as shown by Zhu *et al.* [75]. A number of approaches have been suggested in the literature for determining the optimal CST for a given network. Ma *et al.* [36] propose a solution in which the Access Point (AP) calculates a new CST to be used by all terminals in the network. Haghani *et al.* [21] extend this by proposing that terminals in the network use feedback from the AP to determine their own CST. Finally, Thorpe *et al.* [64] show that an optimal value for the physical carrier sense threshold can be found which maximises aggregate throughput in a network. Their 802.11k Adaptive Physical Carrier Sense algorithm (K-APCS) uses information provided by the radio resource measurement mechanism of

IEEE 802.11k [26] that allows terminals to share detailed radio performance metrics with other terminals in the area.

### 2.2.5 Physical Layer Solutions

Solving the hidden terminal problem has also been attempted at the PHY layer. The main symptom of hidden terminals is the level of collisions. In fact, even if terminals are hidden from one another, a CSMA network can still perform without hidden terminal collisions if those terminals do not happen to transmit at the same time. The problem only exists when terminals attempt to transmit concurrently, causing a collision.

The capture effect [14] occurs when one terminal's colliding transmission is received at a significantly higher signal level than anothers. The louder terminal effectively captures the channel, drowning out the weaker one. Lee *et al.* [33] show that even when hidden terminals cause collisions, one transmitter may be able to capture the channel if its receive SNR is high enough and the receiver hardware is able to resynchronise its receive path. This is known as Message in Message mode or Restart Mode. Restart Mode allows the louder transmitter to have its transmission correctly decoded even in the presence of a weaker colliding frame. While this may reduce the overhead involved in a collision due to the captured frame not needing to be retransmitted, it also introduces medium unfairness.

Gollakota *et al.* propose a new 802.11 receiver design called ZigZag that is able to decode through collisions [20]. By exploiting the fact that hidden terminals generally continue to collide on subsequent retransmissions coupled with the jitter that the 802.11 DCF introduces, ZigZag is able to synchronise its decoder in such a way that allows it to decode both colliding packets. ZigZag is able to provide interference cancellation in situations where the bitrates in use are close to the SNR threshold.

### 2.2.6 Why the hidden terminal problem can't be solved in IEEE 802.11 networks

This chapter has introduced the hidden terminal problem and overviewed the techniques which have been proposed to overcome it. In the case where network operators have an existing IEEE 802.11 network, there is little scope to implement any of the proposed solutions, especially given the cost-sensitive nature of commodity network deployments. To date there is no widely accepted solution to the hidden terminal problem. In general, the approaches suggested in the literature either do not generalise, are not backward compatible or imply extra costs.

Transport layer solutions require the end hosts to be aware of the presence of wireless hops along the forwarding path. While this may be possible in some specific cases such as a self-contained wireless sensor network, it is not possible in the general case where end hosts may not be aware of the wireless network.

MAC layer solutions are either too expensive, for example requiring extra transceivers, or do not provide backward compatibility with existing clients. Consider the case of an operator providing a wireless access network. The provider does not have any control over the client devices which may connect to the network, so a backward compatible solution is imperative if they are to continue to provide access to existing IEEE 802.11 clients.

A physical layer solution such as ZigZag requires retro-fitting hardware into an existing network, which could be considered cost-prohibitive. IEEE 802.11's built in RTS/CTS virtual carrier sense mechanism fails to cope with the hidden terminal problem in many cases. In summary, there is no universally accepted solution to the hidden terminal problem in IEEE 802.11 networks at this time.

## 2.3 Measurement of Hidden Terminals

This chapter has so far shown that the hidden terminal problem is significant and that there are no viable solutions to the hidden terminal problem in an operational IEEE 802.11 network. The ability to detect and measure the hidden terminal problem is therefore important.

The remainder of this thesis focusses not on how to solve the hidden terminal problem but on how to measure it within an existing network. Such a measurement solution would be valuable to network operators who could use the measurement data to better deal with the problem.

This thesis focusses on measuring the hidden terminal problem within an IEEE 802.11 network. More specifically it attempts to do so in a way that does not intrude on the operation of the network it is measuring. It also aims to not require the provision of extra monitoring hardware in order to keep the cost of measurement to a minimum.

In order to ensure that such measurement can be performed on a wide variety of existing IEEE 802.11 networks, this thesis focusses on ensuring measurement can be performed using commodity hardware. The remainder of this chapter presents a description of existing hidden terminal measurement solutions and a discussion of the challenges faced when attempting to measure hidden terminals in a wireless network.

### 2.3.1 Existing Hidden Terminal Measurement Systems

In 2004, Raya *et al.* presented DOMINO [52], a system for detection of greedy behaviour in IEEE 802.11 wireless networks. It does so by using a passive external monitor at the access point (AP) to detect anomalies in client backoff times. They specifically focus on detecting clients with modified 802.11 contention window parameters that allow unfair access to the medium. A benefit of DOMINO is that it does not require modification to the client devices themselves and is an entirely passive system. While the original intent of DOMINO was focussed on detecting greedy client behaviour, the same principle can be used to detect the transmissions of hidden terminals.

The MOJO project [59] aims to diagnose a range of underlying physical layer anomalies that cause problems in wireless networks. It does so by instrumenting client devices with an additional passive sniffer device and software to perform measurement and collate data at the access point. The AP then applies detection algorithms to diagnose physical layer anomalies. In the case of hidden terminal detection, MOJO records the transmission time of each frame from each sender and detects concurrent transmission. While MOJO is able to detect hidden terminals it does so at the cost of additional radio hardware on each client device. This may be feasible if the network operator has control over the client devices, however in the general case it is not.

Li *et al.* [34] propose a combined passive and active technique for detecting hidden terminals. Their approach listens for control frames and detects when a CTS is received without a corresponding RTS, indicating that the intended receiver of the CTS is hidden from the terminal under measurement. This passive method will not always give an accurate count when there is no ongoing background traffic so occasionally a probe frame is injected to solicit a response from each terminal's two-hop neighbours to generate a complete picture of the neighbourhood. This method is simple enough to be implemented within the MAC and appears to be intended for use by a wireless network stack to build a hidden station table so that the MAC can determine if RTS/CTS is necessary for each frame.

### 2.3.2 Measurement Challenges

Both DOMINO and MOJO rely on the use of passive external capture as a method of capturing traffic on the channel. Passive external capture, also known as "vicinity sniffing", is a method for performing wireless link layer capture and analysis. It involves placing "sniffer" boxes in areas that are to be measured [73, 27]. Passive external capture is physically separate from the network being measured - it simply listens on the shared medium and records packet transmissions that the capture point is able to decode. As such it does not introduce any additional overhead to the wireless routers nor does it consume any additional wireless resource. Because of its simplicity, the technique is used extensively in the literature surrounding wireless measurement.

Because of the nature of the wireless medium, the capture point can only provide a picture of the channel conditions at the point where it is placed, which may be significantly different to those at the intended receiver. Frames which are decoded correctly by the capture point may not be decoded correctly by the intended receiver and vice-versa, leading to an inconsistent view of the channel. Multi-path and other environmental effects may cause the recording of the channel conditions to differ between the capture point and the intended receiver. Potentially different RF frontends and antenna configurations also affect the accuracy of the capture.

Schulman, *et al.* [57] investigated the "fidelity" of wireless traces captured using such methods and found that many traces are incomplete or inaccurate, either missing packets or having inaccurate packet timestamps. Their paper highlights the need for caution when using passive external capture as a measurement tool.

Yeo, *et al.* [74] discuss the limits of vicinity sniffing for distributed measurement and suggest that multiple traces, taken by different capture points, should be merged to reduce the problem of individual capture points recording a different set of frames to the network nodes. Such merged traces are of limited use for analysis of the conditions at the link-level as they contain information from physically separate capture points.

Finally, the use of passive external capture requires the provision of extra hardware to the network to act as capture points. This limits its applicability in a scenario where long term wide scale network measurement and monitoring is desired, as the cost of outfitting a network with these extra devices would be prohibitive. As such, any proposed measurement system that is to be deployed by a cost-sensitive network operator must use the capabilities of their existing hardware.

## 2.4 Chapter Summary

This chapter has provided an overview of the hidden terminal problem, why it exists and why it is important. Several attempts have been made to either solve the hidden terminal problem or reduce its effect, though there is currently no widely accepted solution that is applicable to existing IEEE 802.11 networks. For this reason, detection of hidden terminals and measurement of the effect they are having in an operational network is important.

Existing hidden terminal detection and measurement techniques do not scale to the point where they can provide network wide, long-term measurement of large networks. This is especially true in cases where the network operator does not have access to the client devices. Their reliance on passive external capture techniques has been shown to be both costly and unreliable.

This thesis aims to solve the problem of detecting and measuring the hidden terminal problem within operational wireless networks in such a way that does not require the use of passive external capture techniques. In this way, the measurement can be easily deployed and be useful to network operators.

# Chapter 3

# Detecting Hidden Terminals

This thesis proposes that the existence of hidden terminals and their impact on network traffic can be discovered using existing commodity hardware and new software algorithms.

In this chapter, two methods for the detection of hidden terminals are presented. The first method uses network connectivity measurements to identify areas of the network where hidden terminals exist. The second method uses the strict timing requirements of the IEEE 802.11 Distributed Coordination Function (DCF) to detect and quantify the number of collisions caused by the hidden terminals in an area of a network.

The two methods differ in their ability to identify different aspects of the hidden terminal problem. The first method is able to identify problem areas in which hidden terminals exist and hidden terminal collisions may occur. The second method is able to directly measure the effect of the hidden terminals on network performance. The methods can be used together to build a picture of the potential and current impact of hidden terminals in a network.

The methods are described separately in this chapter. First, the network connectivity method is described and it is shown that areas where hidden terminals have the potential to affect network performance can be identified. Second, the timing analysis method is described and it is shown that the effect hidden terminals have on a network can be quantified.

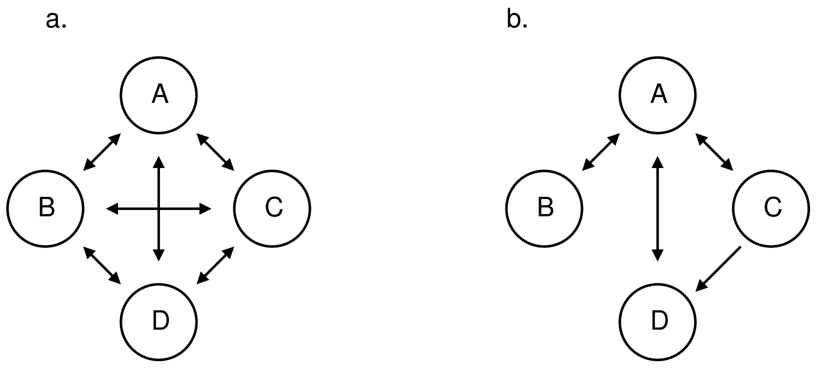

Figure 3.1: Sample connectivity graphs. Each vertex represents a terminal and each edge represents connectivity. A strongly connected graph, such as shown in (a) indicates no hidden terminals. A weakly connected graph, such as shown in (b) shows that terminals $B$ and $D$ are hidden from one another, indicating the potential for hidden terminal collisions at terminal $A$. Additionally, a one-way hidden terminal relationship exists between terminals $C$ and $D$ indicating that $D$ can sense the transmissions of $C$ but not vice-versa.

## 3.1 Network Connectivity

The first method uses connectivity measurements of a network to build a graph of the connections between terminals. Graph theory can then be used to reason about the graph and identify areas where hidden terminals exist.

A directed graph can be used to model the connectedness of a network. In Figure 3.1, each vertex in the connectivity graph indicates a terminal and each directed edge indicates the ability of the vertex at the head of the edge to carrier sense the tail vertex. A strongly connected graph (one where each vertex has an incoming and outgoing edge to each other vertex) would indicate that there are no hidden terminals and CSMA is able to perform effectively.

If the graph is weakly connected, hidden terminals exist. Hidden terminals can be found in this graph by examining vertices and looking for cases where two terminals both have outgoing edges to a common third vertex but not to each other. If the disconnected vertices do not both have outgoing edges to a common third vertex then they are able to concurrently transmit without causing collisions.

In more formal terms, to find if hidden terminals can affect terminal A, find the set of vertices $V_A$ that are a tail vertex to an edge with the head vertex of A. If the subgraph containing the vertices $V_A$ is not strongly connected, the hidden terminal problem has the potential to affect terminal A. For each vertex $v$ in the set $V_A$, $v$ is hidden from any other vertex in $V_A$ that it does not have an outgoing edge to.

### 3.1.1 Problems with measuring connectivity

Measuring the connectedness of a network based on carrier sensing using only commodity wireless hardware is not always possible. Sensing a transmission is not always enough to determine where that transmission came from. The transmission needs to be decoded in order to be able to obtain the transmitter address of the frame.

Sometimes a terminal that is able to sense the carrier of another terminal's transmissions may not be able to decode the packet being received. The frame may have been received at too low a signal to noise ratio (SNR) to decode or it may have been involved in a collision. If a receiver is never able to identify the transmitting terminal from its transmissions but can detect its carrier then the edge from the transmitter to the receiver cannot be added to the graph, even though CSMA can operate to reduce collisions (see Figure 3.2).

Many researchers have attempted to identify the source of transmissions without decoding frames by using Received Signal Strength Indication (RSSI) as an indicator of the distance between the transmitting and receiving terminals. Literature on location determination in wireless networks such as [9, 8, 67] suggests that the use of RSSI as a distance determination metric is inaccurate and that significant calibration and multi-lateration (measuring distance to multiple reference points) work is required to achieve acceptable location determination results. Variation in RSSI measurements can be caused by a number of factors such as multi-path fading, environmental conditions, and transmitter and receiver hardware implementation differences. Calibration and multi-lateration are achievable in indoor environments, however they are not always appropriate in large outdoor environments such as long-distance rural or remote wireless networks.

The use of high precision time-of-flight measurements such as in [47] have been shown to be more reliable, though requires much more accurate timing

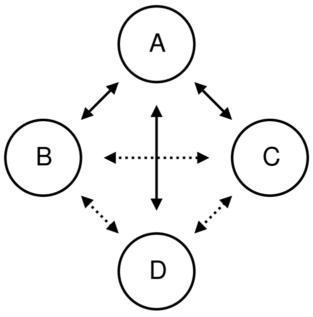

Figure 3.2: In this connectivity graph, full connectivity is shown by solid arrows, whereas the ability to carrier sense but not decode a terminal's transmissions is shown in dashed lines. These lines would not normally appear on the connectivity graph, and so terminals $B$, $C$ and $D$ would appear as hidden from one another when they are in fact not.

measurement than is available on commodity hardware.

By constraining the graph to indicate receive connectivity, rather than carrier sense connectivity, the graph can only indicate the *possibility* of hidden terminals. As a consequence, the connectivity graph shows which nodes are *not* hidden terminals and which nodes *might* be hidden from one another. This is a useful result because it removes the hidden terminal problem from the potential causes of poor performance in many cases.

The hidden terminal problem will only affect the performance of the network if the transmissions of two or more hidden terminals overlap at a receiver. Terminals could potentially remain hidden from one another for long periods of time without affecting the performance of the network if their transmissions do not overlap. This is a function of the traffic patterns of a network. In this case, in graphs based either on carrier sense or receive range, the terminals would be indicated as hidden even though they are not causing a reduction in performance. Because of this, connectivity graphs can only show the potential for hidden terminals to cause problems.

Graphs based on receive range rather than carrier sense range will show a higher potential for hidden terminal problems. It should be noted that if two terminals are not within carrier sense range then they will most certainly not be within receive range, therefore a graph based on receive range will not indicate fewer hidden terminals than a graph based on carrier sense range.

Building a graph that shows receive-level connectivity is a far more practical solution than building a graph that shows carrier sense connectivity. It can

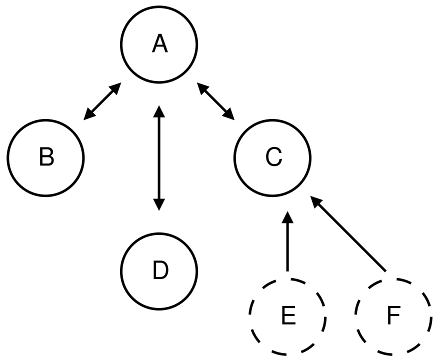

Figure 3.3: In this connectivity graph, terminals $A$, $B$, $C$ and $D$ are under measurement. Terminal $C$ can also detect transmissions from terminals $E$ and $F$, which are *not* under measurement. These *external* terminals are marked with dashed circles, and do not return any connectivity data of their own. As such they appear to be hidden from one another in the connectivity graph but this may not be the case.

be performed using the commodity hardware used for low-cost community wireless networks. Building graphs of carrier sense range in this context is a very difficult problem to solve as there is no practical way to identify the source of a signal without first decoding it.

Another limitation of the connectivity method is that it cannot identify hidden terminals that are not part of the measured network. For example, a measured terminal may be able to detect and decode transmissions from two terminals that are not part of the network. These terminals may be hidden from each other and potentially causing collisions at the measured terminal. However, because it is unknown which terminals the non-measured terminals can hear, the connectivity graph cannot be completed (see Figure 3.3). This is a significant factor in deployments where networks are co-located, such as in urban environments.

### 3.1.2 Summary

Network connectivity graphs are a practical way of determining where the potential for hidden terminal problems may exist in a network. However connectivity graphs cannot directly measure the impact hidden terminals are having on a network. Moreover they cannot include connectivity information from terminals that are not under measurement. Network connectivity measurements can give a network operator valuable information about the operation of their network, for example, network connectivity can rule out hidden terminals as the cause of poor performance if the network connectivity graph is well connected. It can also highlight areas of the network which have the potential to be affected by hidden terminal collisions, even when none are occurring.

## 3.2 Packet Timing Analysis

Hidden terminals can also be detected using receive packet timing. This approach measures the effect hidden terminals are having on the network at the time of measurement, rather than the potential for hidden terminal collisions to occur. The use of receive packet timing and the techniques described in this section are new and novel.

Before describing the approach, this section reviews the IEEE 802.11 PPDU format and the IEEE 802.11 DCF rules that are necessary to understand the method. Next, the different types of collisions that hidden terminals cause are characterised. Finally, a method for detecting transmissions from hidden terminals as well as a method for estimating the overall collision probability due to hidden terminals is presented.

The method is validated with experiments in a controlled laboratory environment which are discussed in Chapter 4. An implementation of the method in a network carrying user traffic and the main characteristics of the implementation are discussed in Chapter 5.

### 3.2.1 The IEEE 802.11 PPDU

The Physical Layer Convergence Protocol (PLCP) allows IEEE 802.11 stations to transfer data between stations while supporting multiple PHY types. The following is a short description of the IEEE 802.11 PLCP Protocol Data Unit (PPDU) which is used to transfer data between IEEE 802.11 stations on the wireless medium.

Figure 3.4 shows how an Ethernet frame generated by a host network stack is converted to an IEEE 802.11 MAC Protocol Data Unit (MPDU) and then

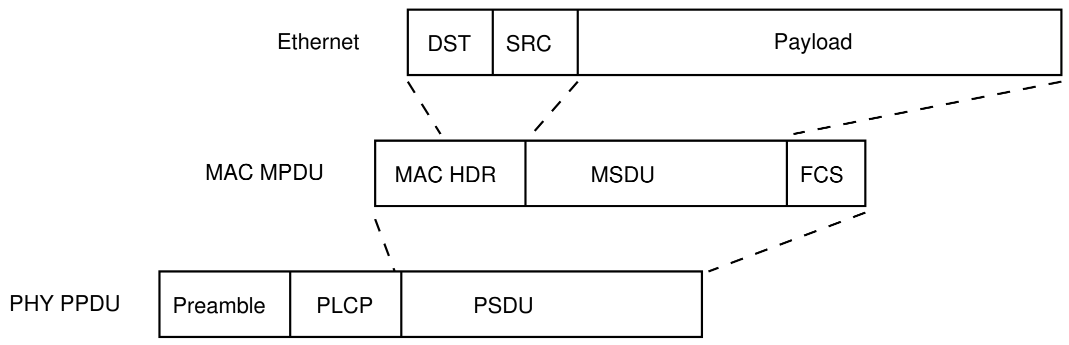

Figure 3.4: The encapsulation of an Ethernet payload within an MPDU and PPDU.

encapsulated within an PPDU. The MPDU contains the standard IEEE 802.11 MAC header followed by the Ethernet payload as the MAC Service Data Unit (MSDU) and a Frame Check Sequence (FCS) covering the MPDU. The MPDU format is independent of the IEEE 802.11 PHY used to transmit it, and is colloquially referred to as a “raw 802.11” frame. In most packet captures, it is the IEEE 802.11 MPDU that is captured and subsequently analysed. However, when the MPDU is transmitted, it is also encapsulated within a PHY layer dependent PPDU which allows 802.11 stations to support multiple PHY types and encoding mechanisms. The PPDU encapsulation and decapsulation is generally hidden from the host operating system as it is a physical layer function performed by the wireless network interface hardware at the physical layer.

The PPDU represents the final bitstream before it is serialised to the medium. It consists of a PLCP preamble, a PLCP header and a PLCP Service Data Unit (PSDU). The PSDU contains the MPDU described above.

The preamble is a bit sequence that enables a receiver to detect the start of a transmission and synchronise its clock to the transmitter so that it may decode the remainder of the frame. The PLCP header is sent at a fixed encoding for each PHY type and encodes the length of the frame in microseconds as well as the encoding used for the PSDU. This allows receivers to correctly decode PSDUs with a range of encoding schemes as well as continue to update the virtual carrier sensing mechanism when an encoding scheme is used that the receiver is unable to decode. Finally the PLCP header contains a CRC field to ensure that it was decoded correctly. If the PLCP header CRC check fails, then the PPDU is discarded.

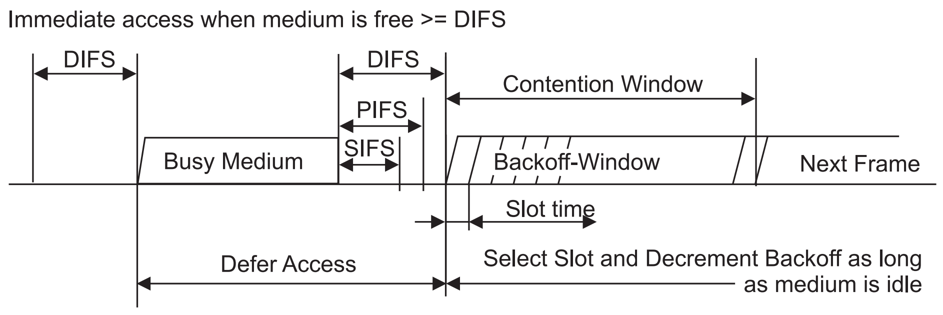
Immediate access when medium is free >= DIFS
DIFS
DIFS
Contention Window
PIFS
SIFS
Busy Medium
Backoff-Window
Next Frame
Slot time
Defer Access
Select Slot and Decrement Backoff as long as medium is idle

Figure 3.5: The IEEE 802.11 basic access procedure, as it appears in the IEEE 802.11 1999 standard.

Each PHY type has its own short and long preamble lengths and PLCP header encoding scheme so the exact format and length of a PPDU depends on the PHY type in use as well as options such as whether the station is sending frames with long or short preambles. For a given packet, the length of time it occupies on the medium is a function of the length of the original Ethernet frame encapsulated within the MPDU and the PHY type, PHY options and the PSDU encoding scheme.

### 3.2.2 The DCF Basic Access Procedure

In order to synchronise the transmissions of multiple terminals, the IEEE 802.11 specification defines the Distributed Co-ordination Function (DCF). The DCF provides a mechanism by which terminals in the network can synchronise their transmissions so as to avoid overlapping transmissions which would cause frame collisions.

The IEEE 802.11 Distributed Co-ordination Function (DCF) is based on the CSMA/CA medium access method, coupled with a set of rules to allow both priority access to the medium as well as backoff in times of medium congestion. In addition to the basic access procedure, RTS/CTS was defined to reduce the effect of hidden terminals. However as has already been shown, RTS/CTS exchanges are not sufficient to completely protect against hidden terminal collisions (see section 2.2.2).

The DCF basic access procedure is shown in Figure 3.5. The inter-frame spacings are defined in the PHY specifications and vary between PHY types. They are referred to here by their names, such as the DIFS, rather than by their absolute values.

When a terminal has a PPDU to transmit, it first checks if the medium has been idle for at least the Distributed Inter-Frame Space (DIFS) period. If it has been, then the terminal may immediately transmit the PPDU without any deferral or backoff. Otherwise, the terminal must defer for the specified inter-frame space for the MPDU that it is trying to deliver.

The shortest inter-frame space is the Short Inter-Frame Space (SIFS) and is used when a terminal has a control frame, such as an ACK, to transmit. If another terminal wanted to transmit a DATA frame during this time, it would defer for the DIFS. The SIFS is shorter than the DIFS so the ACK will be transmitted before the DATA frame. When the terminal waiting to transmit the DATA frame again rechecks the medium (after the DIFS delay) it will find that the medium is busy due to the ACK that started before it, causing it to back off further. This approach gives priority access to the medium for terminals wanting to transmit control frames, ensuring that the DATA-ACK frame exchange is allowed to complete before another exchange begins.

In the case of a terminal wanting to transmit a DATA frame, it must defer until the medium has been idle for the DIFS and then perform an additional random backoff. The terminal picks a random slot in the contention window (CW) and then backs off for this time. If the medium becomes busy during this time, the backoff procedure is repeated once the medium becomes idle. The CW is increased in size exponentially until it reaches an upper limit or until it is reset by the successful completion of the frame exchange. The exponential binary backoff is intended to provide medium stability under congestion conditions.

The DCF relies on carrier sensing to operate correctly. This is either achieved through physical carrier sensing, where the medium is defined to be busy when a received signal level is above a certain threshold, or by virtual carrier sensing, where the terminal gains a knowledge of the network medium's allocation through inspection of the *duration* field in received frames. Each station maintains a Network Allocation Vector (NAV) which records the results of the virtual carrier sensing mechanism. Using a combination of physical carrier sensing and the NAV, the terminal can perform a Clear Channel Assessment

(CCA) to determine if the medium is busy via physical and virtual means and hence participate in the DCF.

When all terminals are able to fully participate in the DCF their transmissions are said to be synchronised. The minimum inter-frame spacing that should be seen on the medium between any two frames is the SIFS period.

When not all terminals are able to participate in the DCF, perhaps due to their inability to carrier sense one another, the transmissions are unsynchronised. That is, they may begin transmission of a PPDU at any given time relative to another existing transmission that may be in progress, possibly leading to frame collisions.

### 3.2.3 Characterising Hidden Terminal Activity

This section characterises the ways in which two unsynchronised terminals may transmit PPDUs relative to each other and the effect this has on the ability of a third terminal, that is within receive range of both, to receive the PPDUs correctly.

Consider two hidden terminals which are transmitting PPDUs randomly in an unsynchronised manner. The two unsynchronised terminals are unable to participate in the DCF, and may transmit in such a way that violates the DCF rules. In this thesis, a *DCF violation* describes any transmission that does not follow the DCF rules. A *collision* is a type of DCF violation that results in the overlapping transmission of two PPDUs.

A third terminal, that is within receive range of the two hidden terminals, will detect each of the hidden terminals' transmissions. The following sections outline the ways in which the hidden terminals transmissions may interact depending on the relative timing and signal strengths of the two frames. To simplify, other errors such as noise caused by bit errors are ignored. Each scenario has an effect on the ability of the third terminal to receive each transmitter's PPDU.

There are six possible interactions in this model: normal operation where no DCF violation occurs; a collision during the frame preamble; a collision during the frame PLCP; a collision during the frame PSDU; a non-destructive preamble collision; and a violation of the SIFS period.

### Normal operation

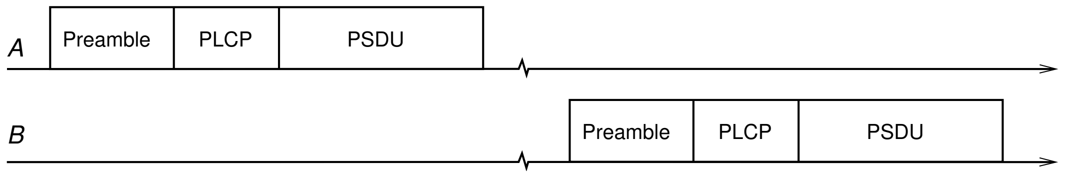

Figure 3.6: Normal operation. Two frames sent by independent terminals do not overlap and do not result in a collision.

While two terminals may be hidden from each other, they only present a problem when their transmissions overlap. Some PPDUs may not overlap (Figure 3.6), and hence not lead to a DCF violation. In this case, in the absence of other problems such as random bit errors, both transmissions will be received correctly by the third terminal. The terminals appear to be operating normally and give no indication that they are hidden from one another.

### Collision during preamble

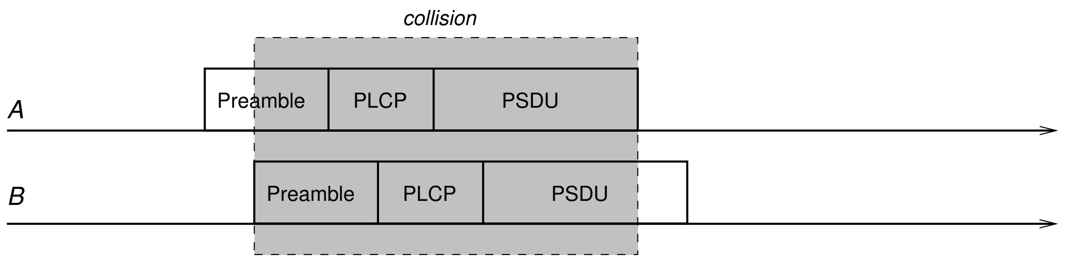

Figure 3.7: Collision during preamble.

On other occasions, a hidden terminal may begin transmission of its PPDU during the preamble of an existing transmission (Figure 3.7). The preamble is used to allow a decoder to synchronise its clock with that of the transmitter and detect the start of a PPDU. If the preamble is corrupted by an overlapping transmission it is likely that the start of the PPDU will not be detected and decoding will not take place.

When concurrent transmissions occur, the radio capture effect may occur [68, 33]. The capture effect allows one terminal's transmission to overwhelm the other if the difference in signal strength is high enough, causing the receiver to "lock on" to the higher powered signal. In this case, the PPDU that is able to capture the channel will be detected and decoded with no dependence on whether it is the first or second PPDU [30]. However, the PPDU that captures the channel may suffer from a higher bit error rate due to the ongoing concurrent transmission of the other PPDU.

If neither PPDU is able to capture the channel, it is likely that neither PPDU will be correctly detected or decoded, leading to both PPDUs being dropped. In the case of a collision during the preamble of a PPDU, at least one of the two PPDUs involved will be dropped, if not both.

### Collision during PLCP header

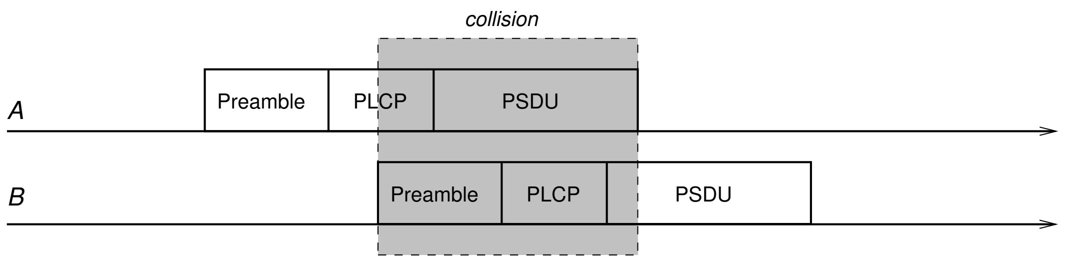

Figure 3.8: Collision during PLCP header.

Sometimes the second terminal may begin transmission during the PLCP header of an existing transmission (Figure 3.8). In this case, the receiver has already synchronised its decoder to the existing transmission and is decoding the PLCP header of the original transmission.

The capture effect may allow the existing PPDU to complete which will lead to the second PPDU being dropped. If the second PPDU captures the channel, the first PPDU will be dropped due to an invalid PLCP header checksum. The second PPDU will often also be dropped as the receiver is unlikely to be able to stop decoding the PLCP header of the first PPDU and resynchronise to the new transmission.

Some vendor-specific hardware extensions, such as the Atheros "Message in Message" mode [56], allow a receiver to resynchronise to a new PPDU that has captured the channel. In this case, the second PPDU may be received.

However the concurrent transmission of the first PPDU may interfere leading to a higher than normal bit error rate.

If neither transmission captures the channel, the existing PPDU will be received, with a higher than normal bit error rate, probably leading to a PLCP header checksum error or an MPDU FCS failure. In this case, the second transmission will not be decoded. In almost all cases, at least one of the two frames will not be received correctly by the receiver.

### Collision during PSDU

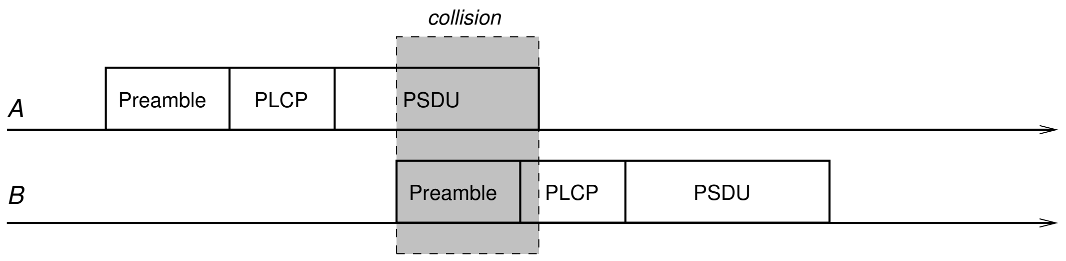

Figure 3.9: Collision during PSDU

In this case, the second terminal begins transmission during the PSDU of an existing transmission (Figure 3.9). The receiver has synchronised with the existing PPDU and decoded the PLCP header correctly and is now decoding the PSDU.

In most respects, the effect of channel capture or increased bit error rate is similar to the previous cases. However, if the second PPDU captures the channel, a truncated MPDU may be passed to the host depending on the implementation of the decoder. Whether the second PPDU is able to be decoded, depends on the implementation of the receiver hardware, so the second PPDU may or may not be dropped.

One or both PPDUs may be dropped or a truncated or highly errored MPDU from the first PPDU may be delivered. The outcome cannot be generalised.

### Non-destructive preamble collision

A special case of the Collision during PSDU occurs if the second terminal begins its transmission slightly before the end of an existing transmission, but

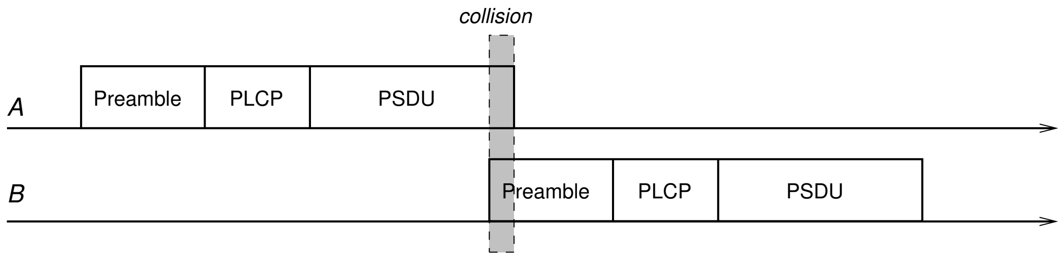

Figure 3.10: Non-destructive preamble collision

not so much that it overlaps the entire preamble (Figure 3.10). If enough of the preamble is uncorrupted, the receiver will synchronise and begin reception of the second transmission after completing the first.

Whether the last few symbols of the first transmission are affected by the colliding preamble is dependent on the relative signal strengths of the two transmissions. If the first PPDU captures the channel, the portion of the preamble that is in collision may not affect the final symbols of the first PSDU. However, if the second PPDU captures the channel, the final symbols of the first PPDU may be corrupted.

Both PPDUs are received, however the first MPDU may contain errors. It is possible though that the first MPDU is received intact. The second MPDU is received correctly if there are no other errors from other sources.

**SIFS Violation**

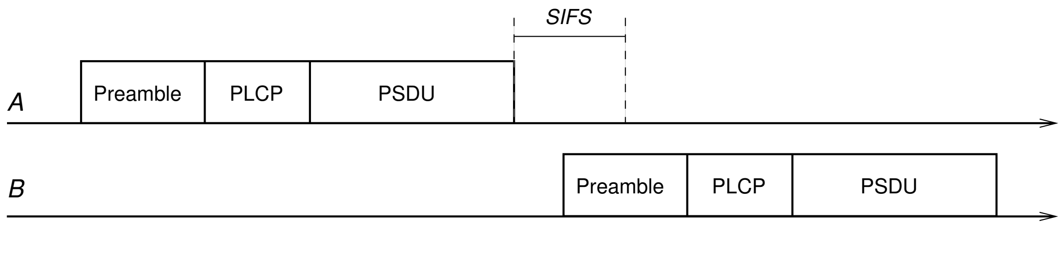

Figure 3.11: SIFS violation.

In each of the previous cases the behaviour of a receiver during a collision is difficult to to predict. The outcome of a collision is dependent on several factors, including the relative signal strengths of the PPDUs involved in the collision (which may lead to the capture effect) and the implementation of the receiver hardware itself.

MPDUs are generally only sent to the host when a PSDU is received. So, it is very difficult for the host operating system to accurately detect collisions occurring. The only collision case described so far in which both packets may be received is the collision during PSDU, and even then the outcome is not guaranteed and one of the MPDUs may be truncated.

The final DCF violation condition that can occur if two terminals are hidden from one another does not involve a collision at all and is distinct from the "normal operation" case presented earlier.

The definition of the DCF states that no transmissions shall start within the SIFS period of the end of a PPDU. In practice, a terminal may begin transmission of a PPDU within the SIFS period if the transmission originates from a terminal that is hidden from it (Figure 3.11). Hence, the second terminal is unknowingly violating the DCF.

If a terminal starts a transmission within the SIFS period and violates the DCF, it indicates that the terminal is not able to detect the already transmitting terminal. An unsynchronised transmission of this type indicates a hidden terminal and is referred to in this thesis as a *SIFS violation*.

While DCF violations that involve collisions are difficult to detect due to the unpredictable nature of the signal strengths of each transmission at a given time, detection of a SIFS violation does not suffer from this issue. A receiver is able to correctly decode and process both PPDUs involved in a SIFS violation and both MPDUs are passed to the host operating system.

### 3.2.4 Using SIFS violations to detect hidden terminals

Hidden terminal activity can be characterised by the DCF violation scenarios presented in the previous section. The host operating system can only reliably detect DCF violations when the two MPDUs involved in the DCF violation are delivered to it. The only scenario in which both MPDUs can be reliably delivered is the *SIFS violation* scenario. All other cases result in a high probability of one or both of the PPDUs being dropped. The outcome of a DCF violation that results in a collision is very difficult to predict due to the channel capture effect, which can be randomly influenced by multipath fading, and the specific vendor hardware extensions that may or may not be present.

It is proposed that observation of DCF violations that result in a *SIFS violation*, rather than a *collision*, can be used to detect the presence of hidden terminals.

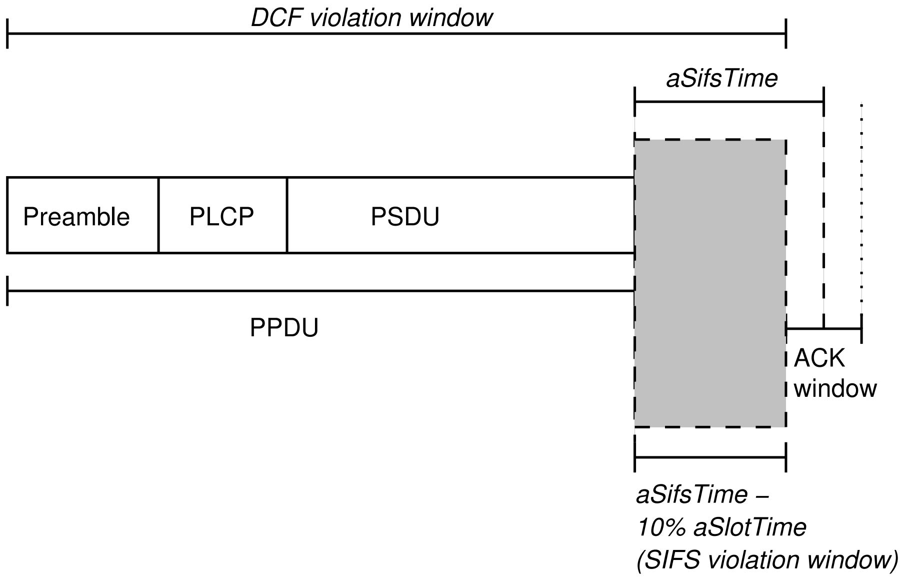

Figure 3.12: The relationships between a PPDU, its ACK window and the SIFS violation window. Any PPDU beginning within the violation window is from a terminal that is unsynchronised with the transmitter of the original PPDU.

Each 802.11 PHY defines a set of PHY characteristics, including the *aSifsTime* which defines the SIFS time for the specific PHY as well as the *aSlotTime* which defines the length of each slot in the contention window. An ACK frame must be received within the ACK window, as defined by the IEEE 802.11 PHY standard:

$$ACKwindow = aSifsTime \pm 10\% aSlotTime \tag{3.1}$$

For example, the 802.11b HR/DSSS PHY defines $aSifsTime = 10\mu s$ and $aSlotTime = 20\mu s$. Therefore the ACK window for an 802.11b PHY is $10\mu s \pm 2\mu s$. If a frame is received within $8\mu s$ of the end of reception of a PPDU it violates the DCF and must be from a terminal that is unsynchronised and therefore the transmitter of the original PPDU is hidden from it. The space following a PPDU in which detection of a transmission indicates a hidden terminal transmission is shown in Figure 3.12 as the SIFS violation window, $v$, and is given by:

$$v = aSifsTime - 10\% aSlotTime \tag{3.2}$$

In the case of a SIFS violation neither of the PPDUs are interfered with as no collision has occurred. As a consequence, the senders of the PPDUs can be identified by inspecting the transmitter address field of the MPDUs involved (assuming that the MPDU type contains a transmitter address and has not suffered from other errors). This allows the method to not only identify that a hidden terminal exists, but also to identify the hidden terminal itself.

Using this method, the transmitter of the first PPDU can be determined to be hidden from the transmitter of the second PPDU. However, the relationship is not necessarily symmetric; the transmitter of the second PPDU is not necessarily hidden from the transmitter of the first. A SIFS violation in which the roles of the two transmitters are reversed must be observed before the relationship becomes symmetric.

### 3.2.5 Using the SIFS violation rate to estimate total collision rate

This section describes a method for using SIFS violations to estimate the total number of collisions occurring due to hidden terminals. It is first described for the simple case of fixed length frames, and then generalised to include arbitrary length frames.

If two terminals are hidden, their unsynchronised transmissions have equal probability of starting at any time relative to one another. Over time, this will cause various types of DCF violations, many of which will be collisions leading to dropped and corrupted PPDUs. However, there is equal probability that a transmission from a hidden terminal will begin within the SIFS violation window, so given enough time, a proportion of DCF violations will be observed as SIFS violations. The length of time before observing a SIFS violation depends on the rate at which PPDUs are being transmitted by each terminal.

Even if both terminals begin synchronised and are transmitting continuously, the relative start times of each PPDU will naturally drift as each terminal has an independent and unsynchronised clock.

As outlined in previous sections, the host operating system is unable to determine the total number of collisions occurring as it only receives MPDUs from PPDUs that have not been dropped at the PHY layer. Neither is it possible to determine the number of collisions occurring by observing the MPDU sequence numbers as these are not incremented upon retransmission, nor does a lost MPDU necessarily indicate a collision (it may have been lost due to fading, for example).

However, the previous section showed that SIFS violations can be detected. Using this information, the total rate of hidden terminal collisions can be estimated.

Consider a single instance of a hidden terminal collision. The first PPDU occupies $t\mu s$ of time on the medium. A second sender must start within $t\mu s$ of the beginning of the first PPDU, i.e, the vulnerable period is $t\mu s$. To detect a SIFS violation, the vulnerable period must be extended by the length of the SIFS violation window, $v$, as defined in equation 3.2. Therefore, the total DCF violation window, $w$, in which a hidden terminal may begin transmission and either cause a collision or SIFS violation is given by:

$$w = t + v \tag{3.3}$$

An independent transmitter causing a DCF violation has a uniform chance of beginning anywhere within the DCF violation window, $w$. When a DCF violation occurs, the proportion $p'$ of those occurring within the SIFS violation window and hence being able to be detected by the host operating system can be given by the equation:

$$p' = \frac{v}{w} \tag{3.4}$$

This gives the probability of a single colliding pair of PPDUs causing a SIFS violation. Consider two hidden terminals, transmitting independently of each other at a fixed rate with fixed length PPDUs. A fixed proportion of PPDU pairs will result in a DCF violation. Of those, a smaller proportion will be observed within the SIFS violation window and the rest will occur within the remainder of the DCF violation window as collisions. Therefore $p'$ gives the overall proportion of SIFS violations which should be expected to occur and hence be detected by the host.

In order to estimate the total rate, $r$, of collisions caused by hidden terminals, the proportion of SIFS violations detected, $d$, over a period of time of the total number of PPDUs received, $n$, is required. This is given by:

$$d = \frac{\text{SIFS violations detected}}{n} \tag{3.5}$$

The number of PPDUs received, $n$, does not include the count of PPDUs dropped due to collisions. The rate of SIFS violations occurring is proportional to the total rate of collisions occurring to hidden terminals. Therefore the total rate of hidden terminal collision, $r$, can be estimated with:

$$r = \frac{d}{p'} \tag{3.6}$$

### 3.2.6 Estimating collision probability for arbitrary packet lengths

The previous section introduced a method for estimating total collision probability based on the rate of observed SIFS violations for the special case of fixed PPDU length. In an operational setting PPDU lengths will vary on a packet by packet basis, not only due to variations in the length of MPDUs, but also due to rate adaptation algorithms switching PSDU encoding schemes per-packet. This section extends the model to account for variable length PPDUs.

With variable length PPDUs the length of the collision window varies, however the length of the SIFS violation window remains constant. As such, the proportion of SIFS violations to collisions is no longer constant. Instead, when estimating the total rate of collisions over a period of time, the total length of the PPDUs received in that time period must be taken into account, while still allowing for the constant SIFS violation window.

Given the transmission time for a set of frames $T = \{t_0, t_1, \cdots t_i\}$; the constant length of the SIFS violation window, $v$; and the number of PPDUs received, $n$, the equation for $p$, the proportion of of SIFS violations to DCF violations is given as:

$$p = \frac{vn}{\sum_{i}^{n} t_i + v} \tag{3.7}$$

When PPDU lengths are constant, as in the previous section, the equation above simplifies to the original equation for $p'$ given by equation 3.4. However, when PPDU lengths are variable, equation 3.7 takes into account the variable PPDU length versus the fixed violation window.

Now that arbitrary PPDU lengths have been taken into account, the total rate of collisions due to hidden terminals can be estimated by substituting $p$ into Equation 3.6 for $p'$. The final equation for $r$ is:

$$r = \frac{\frac{\text{SIFS violations detected}}{n}}{\frac{vn}{\sum_{i}^{n} t_i + v}} \tag{3.8}$$

### 3.2.7 Comparison to existing techniques

The timing method for detecting hidden terminal activity and estimating the total collision rate due to hidden terminals is novel and unique. Its receiver-based approach is fundamentally different to existing techniques for detecting hidden terminals such as MOJO [59] and is more specific than general approaches such as DOMINO [52].

MOJO instruments each client device with a passive external capture point and collates measured frame *transmission* times at a central server. It infers hidden terminals by comparing frame transmission times from various senders and detecting overlapping transmissions. This approach requires additional hardware at every transmitter which is not only costly, but infeasible in environments where transmitters are mobile or transient. Additionally, MOJO requires clock synchronisation of the distributed measurement points and uses the IEEE 802.11 Timer Synchronisation Function (TSF) to achieve this. However, the TSF is only valid within a single Basic Service Set (BSS) so MOJO can only operate within a single BSS.

In contrast, the timing method proposed in this chapter is receiver-based. Hidden terminal activity is detected at any terminal in the network by observing *received* frames at that terminal. Measurement can be performed in a single location to determine the effect hidden terminals are having at that particular location. Transmitters do not require additional hardware or software and timer synchronisation is not required. The timing method can detect hidden terminals whether they are part of the BSS under measurement or not.

DOMINO instruments an Access Point with a passive external monitor. It does not attempt to detect hidden terminals. Instead it infers greedy client behaviour by detecting misbehaviour in client backoff times. The authors of DOMINO mention hidden terminals as a source of interference to their backoff estimator and filter the "false positives" caused by hidden terminals by adjusting their detection thresholds. The timing method presented in this chapter is distinct from DOMINO's general approach in that it can detect hidden terminals as well as estimate the total rate of hidden terminal collisions. DOMINO makes no attempt to detect hidden terminals or estimate their impact.

## 3.3 Chapter Summary

This chapter has presented two complimentary methods for detecting hidden terminals and measuring their impact on the performance of a network. The first method uses distributed measurement to create network connectivity graphs and analysis to detect *potential* hidden terminals by finding areas in which the graph is weakly connected.

The second method uses measurement at a particular receiver and performs timing analysis of received packets to detect violations of the DCF in the form of SIFS violations that indicate hidden terminal activity. Additionally, the method can be used to estimate the *impact* of hidden terminal activity by estimating the total number of collisions occurring due to hidden terminals.

In Chapter 4, the timing method is validated in a laboratory setting. Chapter 5 describes the experience of implementing the methods within a measurement framework that can be used on an operational network, and Chapter 6 shows results from a real-world deployment of the implementations on an operational wireless network carrying customer traffic.

# Chapter 4
# Validation

This chapter presents an experimental validation of the timing analysis method described in Chapter 3. In particular, the goal of this chapter is to investigate the hypothesis that the total rate of collisions occurring due to hidden terminals can be estimated by observing SIFS violations.

Sections 4.1 to 4.4 describe the series of methods that are used in the final section, 4.5, which describes the experimental validation of the timing method. Traffic generation is described in 4.1; PPDU length calculation is described in 4.2; hardware timestamp accuracy is validated in 4.3; and a method for hiding terminals from one another in a laboratory environment is described in 4.4.

Finally in section 4.5, the timing method is validated experimentally by generating a known number of hidden terminal collisions and analysing the empirical results against the theoretical outcomes.

The equipment used in the following experiments is; Atheros 5212 based mini-PCI 802.11abg wireless NICs, Soekris 4526 microcomputers for the transmitting terminals and a Soekris 5501 microcomputer for the receiving terminal. The 5501 is sufficiently powerful to capture all frames sent by the 4526s without any frame drops due to buffer overruns. The Soekris microcomputers run a customised Linux distribution based on Debian Sarge and the 2.6.16 kernel and the MadWiFi Atheros driver for Linux version 0.9.4. This equipment was chosen because it is believed to be typical of that used by wireless ISPs.

## 4.1 Traffic Generation

A small command-line utility, `wgen`, was created for generating raw 802.11 traffic for the purpose of these experiments. `wgen` operates on *monitor mode* interfaces, allowing the generation and transmission of arbitrary IEEE 802.11 MAC frames while avoiding the automatic rate-control algorithms within the driver.

`wgen` generates IEEE 802.11 MAC frames (MPDUs) prepended with a Radiotap header that describes the desired PSDU encoding scheme (bitrate). The number of MPDUs generated, their length, and the data they carry can be varied.

`wgen` will generate a given number of such frames and write them to the raw network interface for transmission using either a fixed inter-packet delay or a delay taken from a uniform random distribution. Each MPDU generated by `wgen` has its destination address set to the ethernet broadcast address to ensure that only a single transmission is attempted (IEEE 802.11 broadcast frames are specified as unreliable and are only transmitted once) and to ensure that no ACKs are generated by receiving IEEE 802.11 terminals.

`wgen` allows for fine-grained control of the traffic being generated and transmitted by each terminal in the experiment. `wgen` can be used to generate traffic patterns with a given channel occupancy rate. If more than one transmitter running `wgen` is used and there are no other transmitters, the number of collisions that occur can be calculated as will be shown in section 4.5.

## 4.2 Calculating PPDU Length

Figure 4.1 shows an IEEE 802.11b HR/DSSS/long PPDU, which encapsulates the MPDU generated by `wgen`. The length of time this PPDU occupies on the channel at a receiver can be given by the *TXTIME* functions found in the 802.11 PHY specifications. The experiments in sections 4.3 to 4.5 are based on the IEEE 802.11b HR/DSSS/long PHY whose *TXTIME* function is defined as:

.

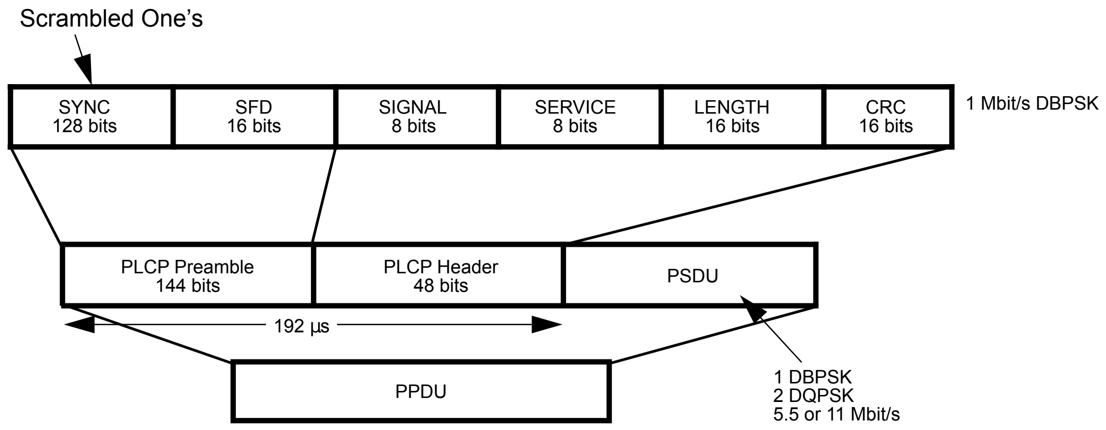

Figure 4.1: An IEEE 802.11 HR/DSSS/long PPDU

$$PreambleLength + PLCPHeaderLength + ceil\left(\frac{LENGTH \times 8}{DATARATE}\right) \quad (4.1)$$

$PreambleLength$ and $PLCPHeaderLength$ are dependent on whether the PPDU is sent with a long or short preamble option. If sent with the short preamble they are $72\mu s$ and $24\mu s$ respectively. If sent with the long preamble they are $144\mu s$ and $48\mu s$ respectively. $LENGTH$ is the MPDU length in octets and $DATARATE$ is the PSDU encoding data rate in units of Mbit/s.

The other 802.11 PHYs such as 802.11a and 802.11g specify their own *TXTIME* functions. In particular they differ as they use OFDM encoding which makes the *TXTIME* function more complex.

## 4.3 Validating RX Timestamps

To calculate the number of SIFS violations, we must know the inter-frame spacing of consecutive PPDUs. This can be calculated if we know the start and end timestamp of each PPDU received. The AR5212 chipset as used by the Atheros miniPCI cards used in these experiments provides a 64 bit timestamp with a $1\mu s$ resolution with each frame received. This section describes an experiment to measure the accuracy of the frame timestamps.

The timestamp provided with each frame describes the timestamp at the *end* of the frame's reception. The first step in determining the accuracy of the timestamp is to calculate the timestamp at the *start* of the PPDU. The start timestamp can be calculated by subtracting the PPDU length (as calculated in section 4.2) from the frame timestamp.

Two experiments were carried out in order to explore the accuracy of the timestamp provided by the hardware. First, the behaviour of the host operating system when generating frames was explored in order to determine if frames could be generated in a controlled manner. This was achieved by setting a single transmitter to generate fixed length PPDUs at a fixed rate. The rate was set slow enough that the hardware transmit queue was always empty when a new frame arrived to be transmitted. An independent passive terminal in monitor mode was used to capture a packet trace and the resulting packet timing was analysed. The results of this experiment are shown in 4.3.1.

Second, the behaviour of the transmitter was explored under conditions where the hardware always had a frame ready to transmit. With host operating system queuing delay effects removed, any packet timing variation would indicate the limits of the hardware when transmitting successive frames. Assuming that the hardware is able to meet the strict DCF timing requirements, any variation from the expected behaviour would likely indicate limited accuracy of the receive timestamps. To achieve this, a transmitter was set to generate fixed length PPDUs at a rate which ensured that the transmitter always had a frame in its hardware transmit queue, ready to be sent. The results of this experiment are shown in 4.3.2.

### 4.3.1 Experiment 1: Transmit queue length = 0

The first experiment used a fixed PPDU length of $12,224\mu s$ (1504 octet MPDU at 1Mbit PSDU encoding with a HR/DSSS/long preamble) and one frame was generated every $50,000\mu s$. With these timings, the median inter-frame spacing should be expected to be $37,776\mu s$, plus or minus any variation introduced by the host operating system. Figure 4.2 shows a timing diagram to illustrate this.

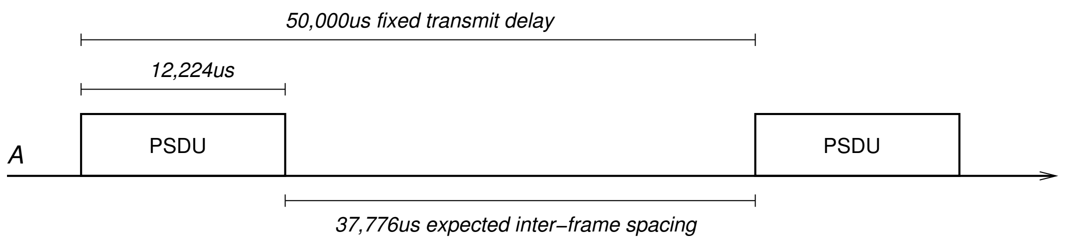
50,000us fixed transmit delay
12,224us
A
PSDU
PSDU
37,776us expected inter–frame spacing

Figure 4.2: Expected inter-frame spacing

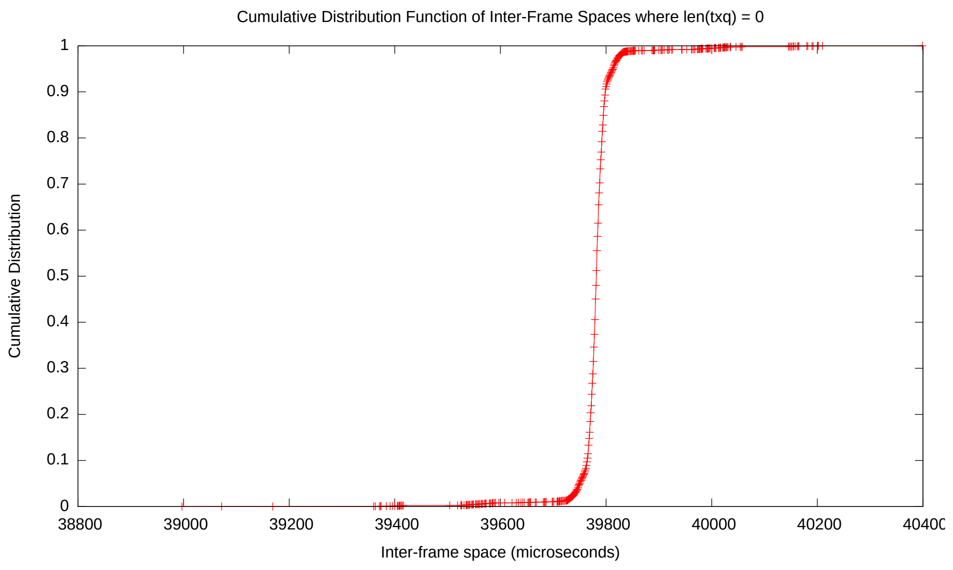

Figure 4.3: Inter-frame spacings from $10,000$ fixed-length PPDUs sent with a fixed delay of $50,000\mu s$.

Figure 4.3 shows the cumulative distribution of inter-frame spacings obtained from the first experiment. The observed median inter-frame spacing was $39,782\mu s$ with a median absolute deviation of $11.86\mu s$.

The variance from the median is most likely caused by variable queueing delay within the operating system. The median inter-frame spacing has a difference to the expected value of approximately $2,000\mu s$. The significant difference from the expected inter-frame space was explored further by modifying the transmitter behaviour and repeating the experiment.

The fixed delay was changed to a uniform random delay with parameters $min = 20,000, max = 60,000$ and the trace was captured once again. Under these conditions the expected cumulative distribution of interframe spaces is a straight line through the range $(20,000 - 60,000)$. Instead, Figure 4.4 shows "steps" $4,000\mu s$ wide. This result suggests that the operating system (most likely as a result of the `select(2)` system call) provides a timer resolution for

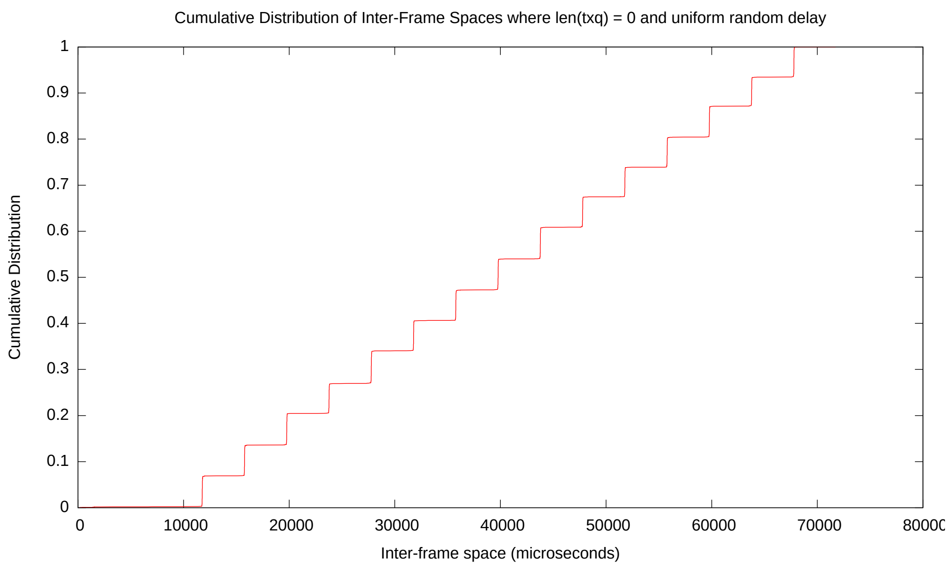

Figure 4.4: Inter-frame spacings from $10,000$ fixed-length PPDUs sent with a uniform random delay with parameters (min$=$20, 000, max$=$60, 000).

packet transmission no lower than $4,000\mu s$.

With a $4,000\mu s$ quantisation effect on packet start time, the original result shown in Figure 4.3 can now be explained. The original delay of $50,000\mu s$ is not a multiple of $4,000\mu s$. The timer resolution causes the frames to be delayed by a further $2,000\mu s$, leading to the $2,000\mu s$ offset from the expected inter-frame spacing.

This first experiment shows that the `select(2)` system call suffers from a $4,000\mu s$ timer granularity, restricting the precision in which frames can be scheduled to be sent from userspace. However, once the transmit timer granularity is taken into account, the receive timestamps appear correct, i.e. the observed median inter-frame spacing matches the expected result, indicating that the operating system is able to generate frames in a controlled manner. The level of variance is within what would be expected by variable queuing delay within the operating system.

### 4.3.2 Experiment 2: Transmit queue length $> 0$

The second experiment examines the transmitter behaviour when host operating system queuing delay effects are removed. By generating frames fast enough that there is always at least one frame in the hardware transmit queue, the hardware always has a frame ready to transmit as soon as it has finished transmitting the current frame. The sender is expected to send a sequence of frames with minimum allowable spacing. This way the distribution of inter-frame spacings on the medium can be examined without any interference from effects of the host operating system, as observed in the previous experiment.

In this experiment, the Clear Channel Assessment (CCA) mechanism is disabled (see the next section, 4.4 for more details). When CCA is disabled, a terminal always senses the medium as idle, even when there are ongoing transmissions. The DCF states that transmission of a DATA frame may not begin until the medium has been sensed idle for at least the DIFS and a Contention Window backoff is performed (see section 3.2.2). In the case where CCA is disabled, the initial DIFS deferral never occurs as the medium is always considered idle. Instead, only the random backoff occurs.

The Contention Window has an initial size of 15 slots. The individual slots are $aSlotTime\ \mu s$ long and in the case of the 802.11b PHY, $aSlotTime = 20\mu s$. A random slot is chosen in the CW each time a back off is performed. Under the conditions of this experiment over a period of time, each slot should be chosen uniformly. The minimum inter-frame spacing expected on the medium (when CCA is disabled and the DIFS deferral never occurs) is $0\mu s$, i.e. the 0th slot in the Contention Window. Practically, however, processing delays in the MAC and PHY mean that even if slot 0 were selected, an inter-frame space of $0\mu s$ is unlikely to be achieved.

Similar to the previous experiment, frames were generated by the `wgen` tool. 1504 octet MPDUs ($12,224\mu s$ PPDUs) were sent with a fixed delay of $12,000\mu s$ in order to ensure that there was always a PPDU in the hardware queue to transmit. Figure 4.5 shows the cumulative distribution of inter-frame spacings as measured by the monitoring terminal.

A step function is shown in the data. These steps reflect the expected operation of the 802.11 Contention Window. Each "step" in the graph is $20\mu s$ wide and represents a single slot in the Contention Window. The consistency of the step widths illustrates that the receive timestamp accuracy is stable. Figure 4.6 shows a single step in the graph. The median absolute deviation for each

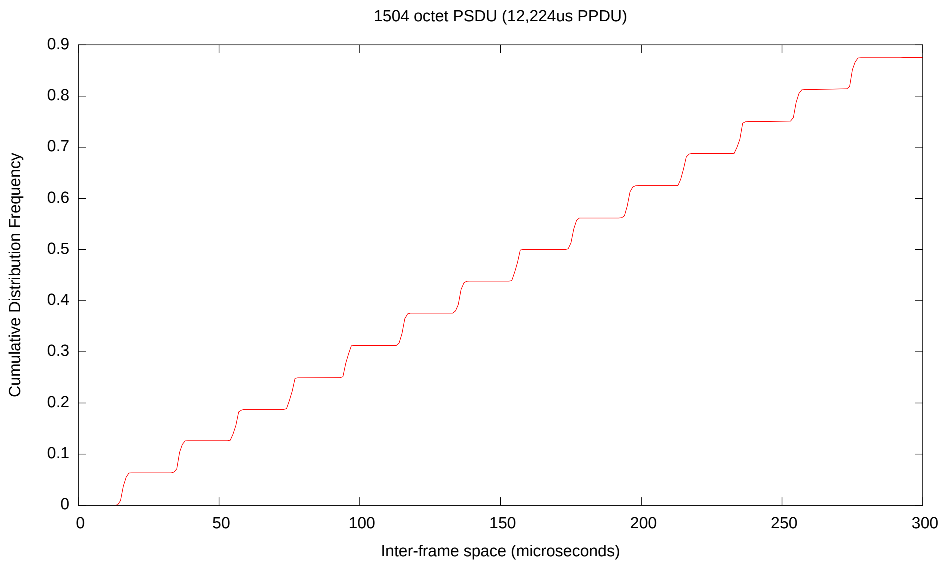

Figure 4.5: Inter-frame spacings from $10,000$ fixed-length PPDUs sent with a fixed delay of $12,000\mu s$. The “steps” reflect slots in the 802.11 Contention Window.

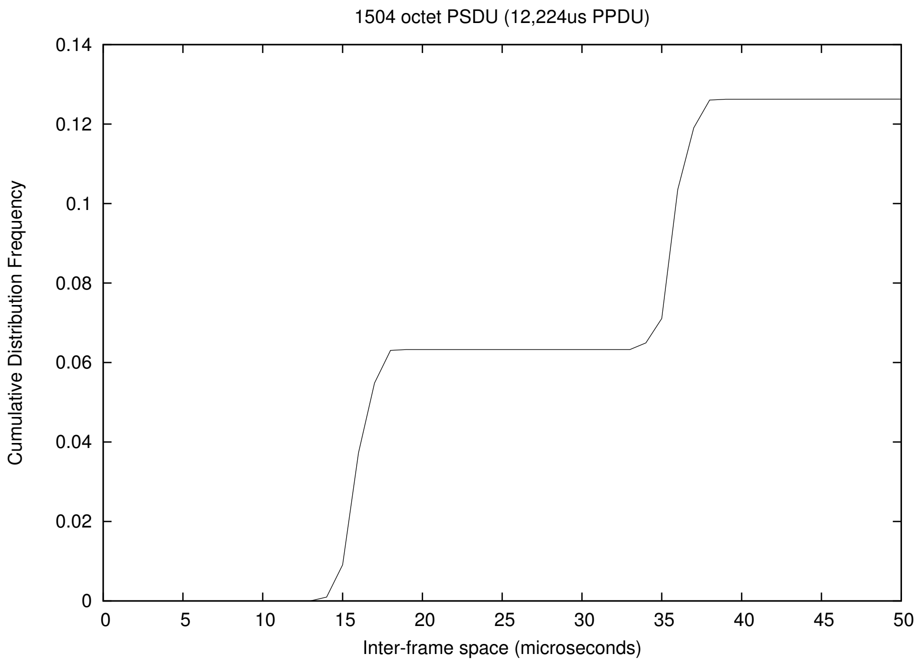

Figure 4.6: A view of the first two steps of the inter-frame spacings from $10,000$ fixed-length PPDUs sent with a fixed delay of $12,000\mu s$. The first “step” has a median of $16\mu s$ with a median absolute deviation of $1.48\mu s$.

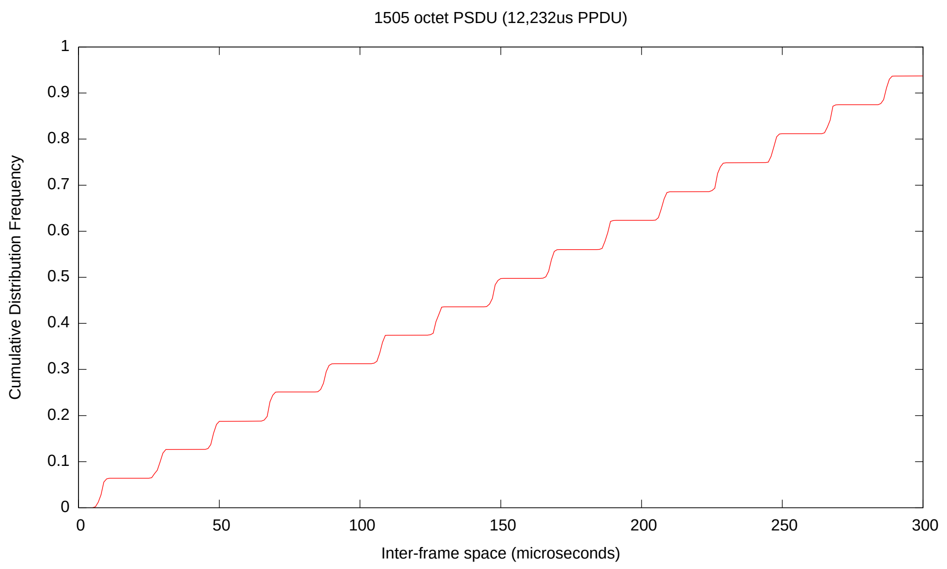

Figure 4.7: Inter-frame spacings from $10,000$ fixed-length PPDUs sent with a fixed delay of $12,000\mu s$. In this case, the PPDUs are $8\mu s$ longer than the previous experiment and the minimum inter-frame space has moved to $8\mu s$.

step is $1.5\mu s$. This indicates that the receive timestamp is accurate enough to be used by the timing method to detect transmissions within the SIFS violation window.

It was expected that the minimum frame spacing would be approximately $1\mu s$ due to the lack of DIFS deferral. However, the results indicate a minimum inter-frame spacing of $16\mu s$. This result was explored further by modifying the PPDU length.

The length of the PPDU was increased by $8\mu s$ by adding one octet to the length of the MPDU. Figure 4.7 shows the result of the same experiment with a longer PPDU. The same slot effect can be seen, however the minimum inter-frame spacing shifts to $8\mu s$. The shift in minimum inter-frame spacing follows a sawtooth pattern as the PPDU length is adjusted, the effect of which can be seen in Figure 4.9, which shows the minimum inter-frame spacing measured for a number of PPDU lengths. Given the consistency of the slot width measurements, it is unlikely that this is a measurement error at the receiver.

Instead, the shifting minimum inter-frame space can be explained with a closer look at the transmitter behaviour. When a transmitter has a frame to send, it defers for the DIFS and then selects a slot in the Contention Window. Slot times however are clocked in a separate time domain to the initial DIFS

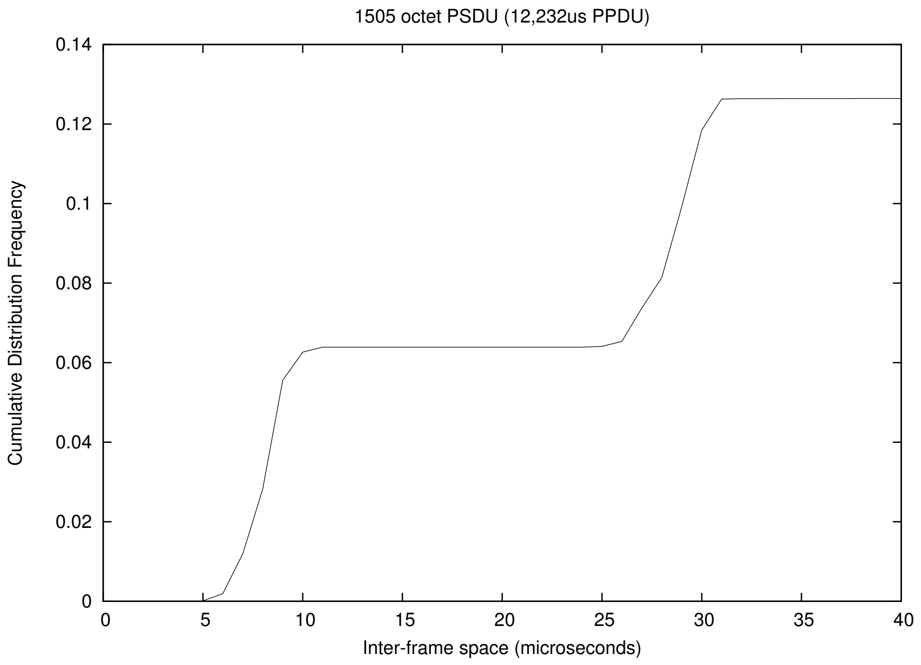

Figure 4.8: A view of the first two steps of the inter-frame spacings from $10,000$ fixed-length PPDUs sent with a fixed delay of $12,000\mu s$. The first "step" has a median of $8\mu s$ and a similar median absolute deviation.

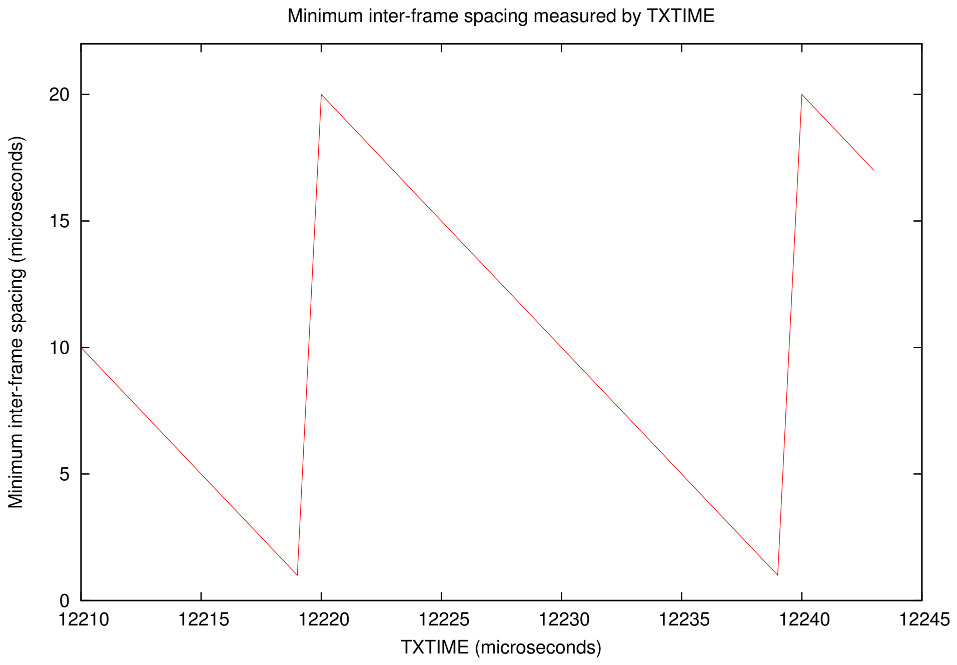

Figure 4.9: The minimum inter-frame spacing measured as TXTIME was increased by adjusting PPDU length.

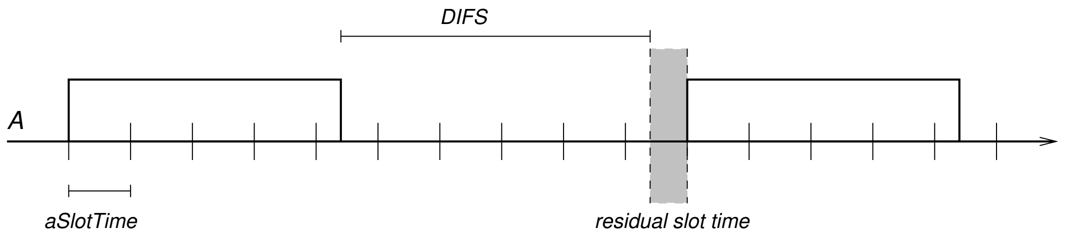

Figure 4.10: Residual slot time caused by clock domain crossing. The PPDU starts on a slot boundary however its duration is not necessarily a multiple of $aSlotTime$. The DIFS deferral starts from the end of reception, however, because the slots are clocked in a different time domain, there exists a residual slot time after the end of the DIFS and before the next slot begins. The length of this residual slot time will vary depending on the duration of the PPDU.

deferral. This clock domain crossing means that there will be some residual slot time left at the end of the DIFS deferral. In this experiment, because CCA is disabled, the residual slot time is left at the end of the previous PPDU rather than the DIFS deferral, however the principle remains the same. This is illustrated in Figure 4.10. The effect of this clock domain crossing is that a transmission may be deferred by up to the length of a slot time greater than would be expected if using a simple, single clock domain model.

As a PPDU always starts on the boundary of a Contention Window slot and the time a PPDU will occupy the channel can be calculated, the length of the residual slot time can also be calculated. The residual slot time is given by:

$$r = aSlotTime - (TXTIME \mod aSlotTime) \tag{4.2}$$

By updating the simple, single clock domain model to take into account separate clock domains for the DIFS deferral and CW slots, the variable minimum frame spacing observed in Figure 4.7 can be explained. When the residual slot time is subtracted from the observed inter-frame spacings, the expected result of a constant minimum inter-frame space is observed. For example, the residual slot time for a $12,224\mu s$ PPDU is $16\mu s$ and minimum median inter-frame spacing observed on the medium was $16\mu s$. The residual slot time for a $12,232\mu s$ PPDU is $8\mu s$ and the minimum median inter-frame spacing observed on the medium was $8\mu s$.

While taking clock domain crossing into account is necessary to adjust for the more complex transmitter model when searching for the minimum inter-frame spacing to validate the correctness of the timestamps, it is not necessary to adjust for clock domain crossing in the general case. The receive timestamp is still an accurate representation of when the frame was received on the channel and hence can be used to determine accurate inter-frame spacings. The clock domain crossing affects transmitter behaviour; it does not affect the accuracy of the received PPDU timestamp.

The results of these experiments have shown that frames can be generated in a controlled manner and that the timestamps reported by the AR5212 hardware are accurate enough (by the result of the median absolute deviation of $1.48\mu s$ when measuring Contention Window slot boundaries) to be used by the timing method to detect SIFS violations.

## 4.4 Hiding Terminals

To measure hidden terminal collisions in a laboratory requires the ability to hide one terminal from another even though they are physically close to each other. The ability to hide terminals from one another is not a requirement of the timing method; it is a requirement of the validation of the timing method in a laboratory environment.

In the experiment, the two transmitting terminals must be hidden from one another so that their transmissions become unsynchronised and concurrent transmission occur. In an indoor environment and in the absence of specialised equipment such as Faraday cages, this is difficult to achieve reliably through physical separation of the terminals while still maintaining connectivity to a common terminal for monitoring.

The method to disable Clear Channel Assessment (CCA) and hence hide terminals from one another in this experiment is described by Anderson, *et. al* in [5]. They identify the registers which affect the operation of CCA in the Atheros AR52XX-based wireless NICs that are used in this thesis. In order to disable CCA, the open-source MadWiFi Linux driver was modified to set these registers accordingly. In particular the `AR5K_AR5212_DIAG_SW` (`0x8048`) register is bitwise ORed with the values `AR5K_AR5212_DIAG_SW_IGNOREPHYCS`

(`0x00100000`) and `AR5K_AR5212_DIAG_SW_IGNORENAV` (`0x00200000`) which disables both physical and virtual carrier sensing.

The two terminals under test were configured with CCA mechanisms disabled. This means that the terminals do not check for the presence of existing transmissions on the medium before transmitting themselves, which causes them to behave in the same way as if they were hidden from one another through physical separation.

Under these conditions and given suitable traffic patterns, over time, DCF violations in the form of collisions and SIFS violations will occur.

### 4.4.1 Validating CCA Disable

The effectiveness of disabling CCA was confirmed experimentally by observing the transmission behaviour of two terminals with CCA enabled and disabled.

`wgen` was used to generate traffic on two terminals with CCA enabled such that they would both have PPDUs to transmit at the same time, leading to channel contention. A third passive terminal was used to capture the resulting traces. The procedure was repeated with CCA disabled. The results are shown in Figure 4.11 by plotting the cumulative distributions of the inter-frame spacings in each trace.

The CCA enabled and disabled cumulative distributions are very different from one another. With CCA enabled, the transmission behaviour follows that of two synchronised terminals competing for the medium. With CCA disabled, the transmission behaviour follows that of two hidden terminals transmitting frames without regard to existing transmissions. This can be seen by the near uniform distribution of inter-frame spacings in the CCA disabled data set.

A detailed view of the data set is shown in Figure 4.12. This shows the behaviour of the two cases where the inter-frame spacing is very short, that is, when the two terminals are competing for the channel.

When CCA is enabled, the inter-frame spacing indicates that the terminals are obeying the DCF by deferring their transmissions and participating in the

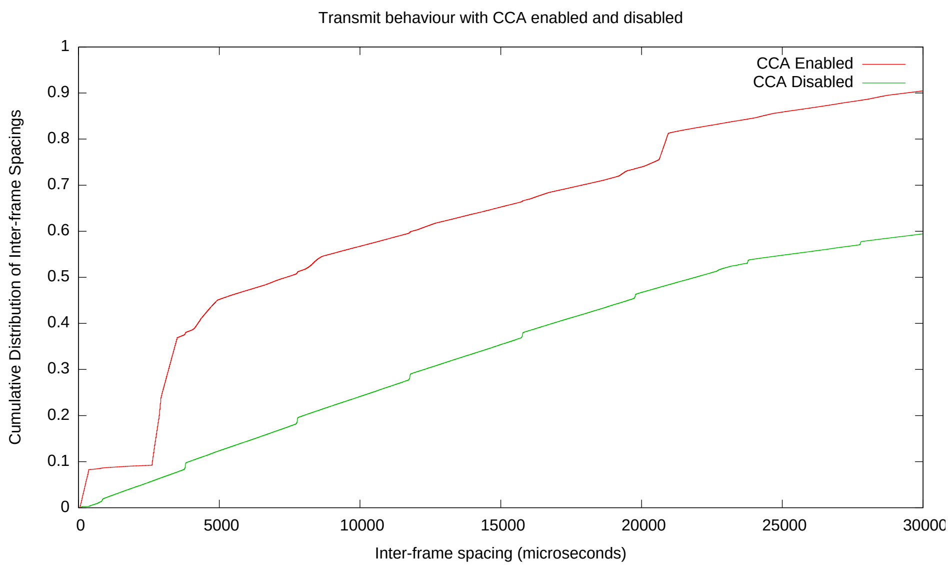

Figure 4.11: Transmitter behaviour with CCA enabled and disabled

random backoff rules. When CCA is disabled, the distribution looks different. A small number of negative inter-frame spacings indicate the occurrence of *non-destructive preamble collisions* as discussed in section 3.2.3. A number of frames appear within the SIFS violation window and the remainder follow the uniform distribution expected from two unsynchronised senders.

These results show that disabling CCA, by setting the debug registers on the AR5212 hardware, allows terminals to be hidden from one another reliably. When the terminals are hidden their transmissions become unsynchronised and SIFS violations are observed.

## 4.5 Experimental Validation of the Timing Method

This section presents experiments carried out in order to validate the timing method presented in Chapter 3.

### 4.5.1 Hypothesis

This experiment explores the hypothesis that the number of PPDUs that start within the SIFS violation window of the previous PPDU can be used to estimate the total number of collisions caused by hidden terminals.

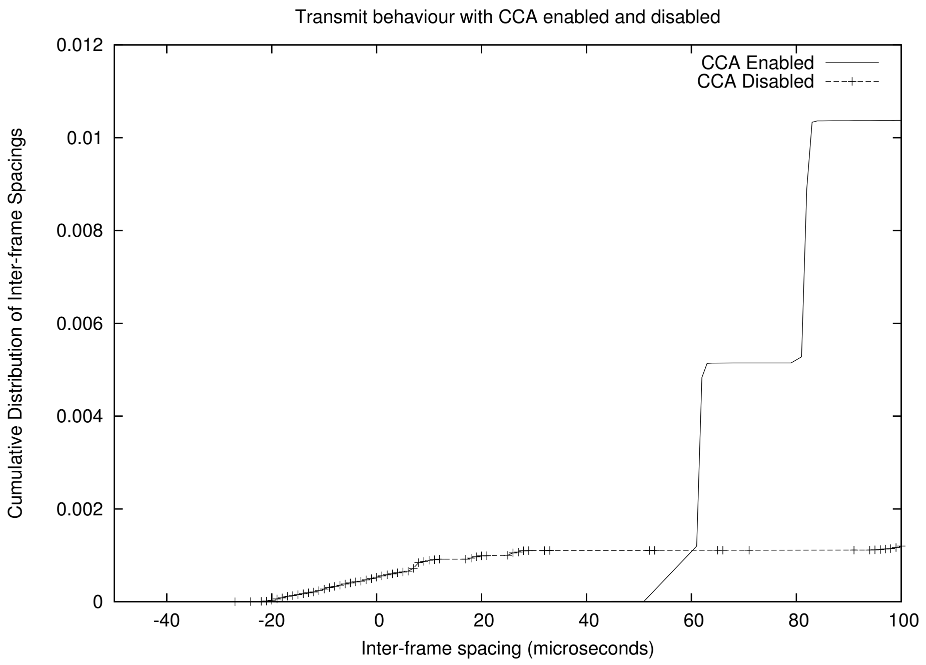

Figure 4.12: Detail view of transmitter behaviour with CCA enabled and disabled. This shows with CCA enabled, the transmitters behave as expected. With CCA disabled, the transmitters behave as if they are hidden from one another. Additionally, a number of PPDUs arrive "before" the end of the previous PPDU (indicating a non-destructive preamble collision) and a number arrive in the SIFS violation window.

Hidden terminal collisions can be generated by hiding two terminals from one another and generating known traffic on each. The total number of collisions can be calculated from the number of frames sent and the number of frames received by a third, passive terminal that listens for frames but does not transmit. If the passive terminal also records the number of SIFS violations detected, the timing method can be used to *estimate* the total number of collisions occurring due to hidden terminals.

This experiment sets out to validate the timing method's *estimate* of the total number of collisions by comparing it to the *known* number of collisions caused during the experiment. The known number of collisions caused during the experiment is controlled by altering the rate at which the two terminals transmit PPDUs.

### 4.5.2 Method

Two identical Soekris 4526 microcomputers were set up in an otherwise quiet radio environment with a modified version of the MadWiFi Linux driver for

Table 4.1: wgen fixed experiment parameters

| Total TX PPDUs | Fixed Delay | Uniform Delay min | Uniform Delay max |
|---|---|---|---|
| $3,600,000$ | $48,000\mu s$ | $1\mu s$ | $90,000\mu s$ |

Table 4.2: Variable parameters

| MPDU octets | PSDU bitrate | PPDU length | Collision Rate |
|---|---|---|---|
| 39 | 5.5 Mbits | $249\mu s$ | 0.75 % |
| 39 | 1 Mbits | $504\mu s$ | 1.2 % |
| 98 | 2 Mbits | $584\mu s$ | 1.4 % |
| 98 | 1 Mbits | $976\mu s$ | 2.2 % |
| 216 | 1 Mbits | $1,920\mu s$ | 4.2 % |
| 452 | 1 Mbits | $3,808\mu s$ | 8.2 % |
| 924 | 1 Mbits | $7,584\mu s$ | 19.8 % |
| 1504 | 1 Mbits | $12,224\mu s$ | 35.4 % |

Atheros based wireless NICs which allowed the terminals to be hidden from one another, as described in section 4.4. A third terminal, a higher powered Soekris 5501, was set up to passively capture all frames on the air and was placed such that the received signal strength from each of the transmitting terminals was approximately equal. The monitor terminal captured frames from both transmitting terminals and recorded them to a PCAP network trace file.

The `wgen` tool was used to generate traffic from each of the two transmitting terminals in such a way that ensured some proportion of PPDU transmissions would cause DCF violations.

A series of experiments were run to generate different traffic profiles from which an approximate collision rate could be calculated. Each experimental setup was run for approximately 24 hours with a fixed inter-frame delay on one terminal and a uniform random delay on the other. Different collision rates were generated by varying the MPDU length and PSDU encoding rate. The fixed parameters for the experiments are given in Table 4.1. The parameters that were varied are given in Table 4.2.

The resulting trace files were analysed and the total number of collisions determined by subtracting the number of frames captured from the known number of frames transmitted. PPDU drops were unlikely to be caused by other factors such as path attenuation due to the proximity of the monitoring terminal to the two transmitting terminals. The monitoring terminal had sufficient CPU power to capture all frames generated by both transmitting terminals while writing the trace to disk without dropping frames. As such, any frame missing in the resulting trace was considered as being caused by a collision with another PPDU.

Frames in which an FCS error occurred were also treated as the result of a collision. With the two transmitting terminals so close together the received signal level was high enough to ensure a negligible bit error rate. As such, all FCS errors were treated as being caused by a second frame starting during the PSDU of an existing transmission.

The proportion of SIFS violations expected to occur within each experiment was determined based on the known traffic profile of each experiment by calculating the SIFS violation window size as a proportion of the PPDU length, as outlined in section 3.2.4.

The number of SIFS violations that occurred within each trace file was determined by calculating the spacing between each consecutive frame pair. Frame pairs in which the spacing was in the range ($0\mu s < x < 8\mu s$) and in which the first frame of the pair suffered no damage were determined to be SIFS violations. The total number of hidden terminal collisions could then be estimated using the timing method (section 3.2) and compared to the known number of frame collisions.

### 4.5.3 Results

Figure 4.13 compares the expected proportion of SIFS violations to hidden terminal collisions by PPDU length to the observed proportion. The expected proportion is calculated using the method outlined in section 3.2.4. The actual measured proportion of SIFS violations to known collisions in the captured trace is plotted alongside and it can be seen that the measured relationship follows the expected result.

The number of SIFS violations can be measured and given the total known number of collisions in the trace, the proportion of SIFS violations to collisions

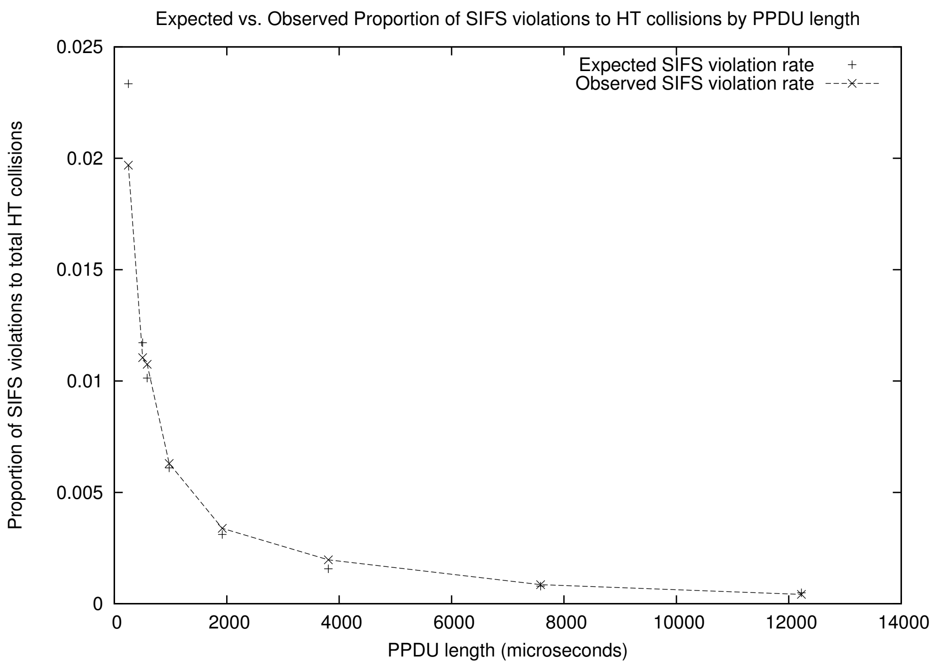

Figure 4.13: Expected versus observed proportions of SIFS violations to hidden terminal collisions by PPDU length.

can be determined. In practice, the total number of collisions cannot be known in advance. Instead, the total number of collisions must be estimated using the number of SIFS violations detected and the number and length of PPDUs received.

By estimating the total number of collisions based on the rate of SIFS violations, the total rate of dropped PPDUs can be estimated, as presented in 3.2.5. These estimates can then be compared to the known number of collisions in each experiment, validating the method.

The method requires the rate of SIFS violations to be measured over time. In order to determine a reasonable length of time before the estimations become stable, the trace data from each 24 hour experiment was split into bins of varying length; 300, 600, 1800 and 7200 seconds. The results from each bin size were analysed separately in order to determine the time period that minimised overall measurement variation.

Figures 4.13(a)–(d) show the *estimated* rate of DCF violation caused by hidden terminal collisions (Y-axis) for each of the four time periods. The binning process allows us to make an estimate for each bin-length sample from the trace data and compile quartile plots of these estimates. The data is plotted

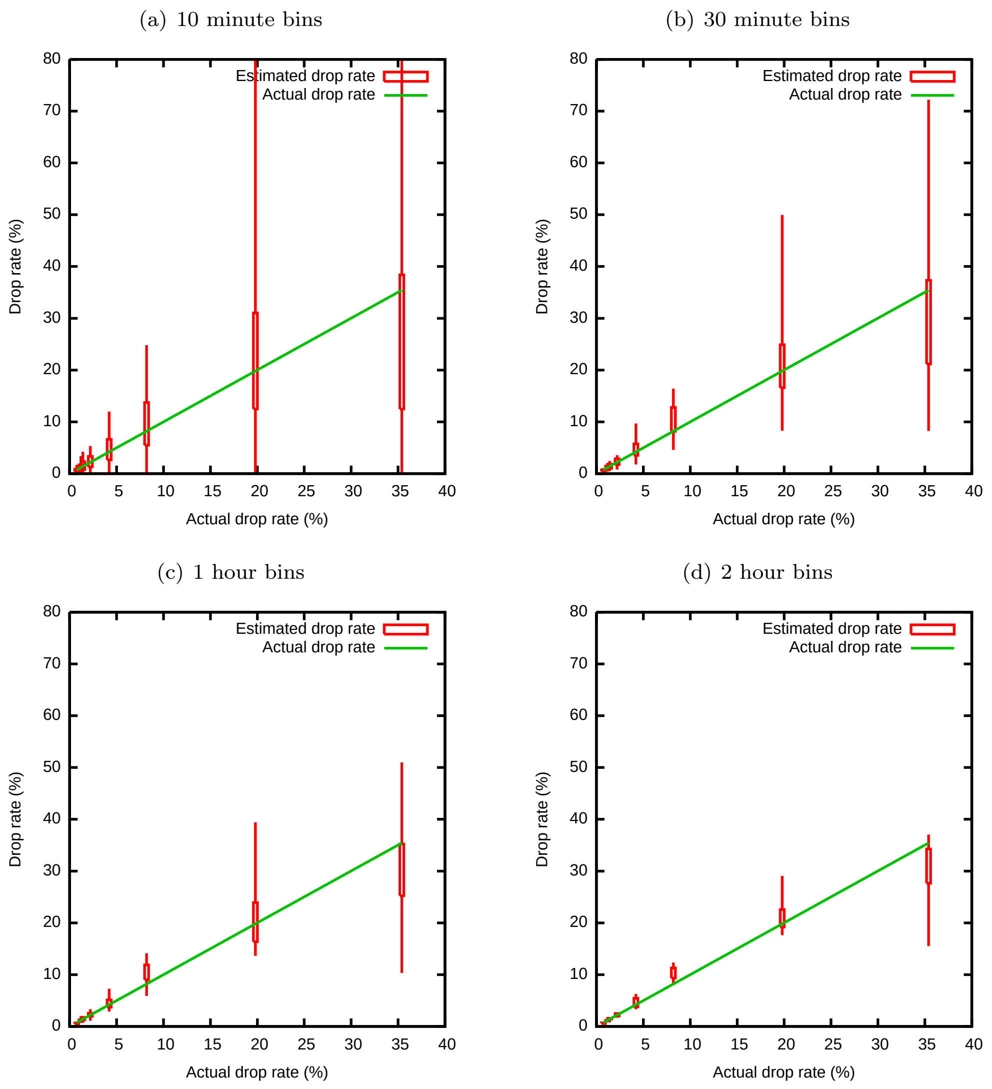

Figure 4.14: Estimated versus Actual drop rate due to hidden terminal collisions.

against the *known* drop rate for each trace (X-axis), which allows comparison of the estimation to the known, controlled drop rate.

10 minute intervals were not sufficient to estimate the drop rate reliably, however the median drop rates estimated are close to the known drop rate, at least where the drop rate is $< 10\%$. By 30 minutes, estimations at the low end of the drop rate range ($< 5\%$) have lower variability, however estimation during times of heavy loss are still very unstable.

Using 1 hour bins, the timing method can achieve an estimation accuracy for loss rates $< 10\%$ of $\pm 5\%$. 2 hour bins improve this accuracy to $\pm 2\%$. For loss rates $> 10\%$, estimation accuracy suffers. This is mainly due to the change in ratios between SIFS violations and collisions. As the drop rate increases, the ratio of SIFS violations (which are detectable) to DCF violations decreases, which causes the timing method to struggle to accurately estimate drop rate.

The 1 and 2 hour bins improve the stability of the estimation at times of heavy loss, however the variation at these times is still very high. A PPDU loss rate greater than 10% is cause for concern to a network operator, however once the loss rate reaches this level it is not necessarily useful to be able to predict the exact level with fine grained accuracy. It is often enough to say that an extreme level of loss is occurring.

This reduction in estimation accuracy at higher levels of loss can in part be explained by the variable $n$ in Equation 3.8. As frame loss rate increases the host is unable to accurately count $n$, the number of PPDUs transmitted. This causes the equation to over-estimate the proportion of SIFS violations detected to frames on the channel and hence affects the overall estimation of DCF violations caused by hidden terminals.

## 4.6 Conclusions

This chapter has investigated the timing method outlined in Chapter 3 and has provided an experimental validation of its efficacy in a controlled laboratory environment.

Experiments in a controlled setting were carried out to validate the accuracy of the receive timestamps provided by the wireless NICs, a method for arbitrarily hiding terminals from one another was confirmed and experiments were carried out that showed that the SIFS violation rate can be predicted based on PPDU length and that the actual SIFS violation rate can be used to predict collision rates due to hidden terminal.

The results of these experiments show that the rate of SIFS violations can be used to estimate the total number of DCF violations (and hence total number of collisions) occurring due to hidden terminals. At low hidden terminal collision rates, the estimation is accurate enough to provide a clear view of the loss occurring due to hidden terminal collisions. However, as the loss rate increases, the estimation accuracy decreases.

The timing method can therefore be used to estimate the rate of collisions being caused by hidden terminals when the overall collision rate is $< 10\%$. For higher collision rates, the timing method can only indicate that high numbers of collisions are occurring – it cannot provide an accurate estimate at these high collision rates.

The next chapter presents a practical implementation of both the network connectivity method and the timing method for deployment on a real-world network. Implementing the method for use on a production network poses several additional challenges which are addressed in the next chapter. Chapter 6 goes on to present results from a deployment of the methods on an operational network carrying customer traffic and shows that the methods are useful in a practical setting.

# Chapter 5

# Implementation

Two methods for the detection and measurement of the hidden terminal problem were presented in Chapter 3 and the methods were validated in a laboratory setting in Chapter 4.

The goal of this chapter is to demonstrate the implementation of the methods within a measurement system that allows deployment across the entirety of an operational wireless network. Network operators can then use the measurement system to detect and measure the effect of hidden terminals over time as an integral part of their existing network measurement and monitoring infrastructure.

This chapter begins by outlining the requirements of such an implementation in section 5.1. These requirements are motivated by the goal of deployment on an operational network. Section 5.2 discusses existing techniques for performing measurement and section 5.3 introduces the WMP Kernel Measurement Framework for performing wireless measurement directly on hosts. Finally, sections 5.4 and 5.5 discuss the implementation of the hidden terminal detection methods within the WMP framework and section 5.6 concludes this chapter.

## 5.1 Requirements for wide-scale implementation

The hidden terminal detection methods aim to provide network operators with useful tools to detect and measure the effect of hidden terminals on their networks. An implementation of the hidden terminal detection methods needs to adhere to a number of specific requirements in order to be useful in an operational setting. This section outlines those requirements which guide the implementation of the methods in the following sections.

### 5.1.1 Scalability

An implementation should scale in its deployment across large networks. In particular, the network connectivity method requires connectivity information from as many participants in the network as possible for it to be effective. As such, any implementation of the hidden terminal detection methods should be able to be deployed across as many of the network hosts as possible without increasing the complexity of the deployment.

### 5.1.2 Accurately capture information

An implementation should accurately capture information. The timing method requires accurate timestamps for each frame and requires all frames to be captured, not just those that are destined for the Basic Service Set (BSS) that the Station (STA) is a part of. Any implementation that cannot provide the necessary timestamp accuracy or which compromises the completeness of the captured dataset is not suitable.

### 5.1.3 Limit adverse affects on the network

An implementation should not impose significant extra overhead on the network by consuming undue amounts of network resource. This is especially important for any implementation of distributed measurement where measurement traffic should not overwhelm legitimate network management or user traffic. Additionally, an implementation should not slow down packet forwarding significantly or negatively affect the performance of the network as perceived by end users.

### 5.1.4 Limit cost

An implementation should not impose significant per-host costs. This is a direct consequence of the requirement for scalability as well as the cost-sensitive nature of operators of networks based on commodity hardware. If the implementation imposes significant per-host costs or if the cost and complexity of deployment does not scale gracefully with the number of measured hosts, then the ability to deploy the implementation across the network is compromised.

## 5.2 Existing Measurement Techniques

This section outlines existing passive measurement techniques that are common in the literature and discusses their ability to meet the requirements set out in section 5.1.

### 5.2.1 Passive External Capture

Passive external capture is one of the most common forms of wireless link-layer measurement likely due to its simplicity. It is sometimes referred to as *vicinity sniffing* [27] or *indirect capture* [7] as it involves sniffing frames in the vicinity of a passive monitor which is not part of the network being measured. For example, a laptop computer could be used to passively sniff frames from a network operating in the same area. This technique is used extensively by wireless monitoring software systems such as Kismet [2] and much of the literature on wireless network monitoring uses this technique. For example, Yeo [73] and Jardosh [27] use passive external capture when characterising indoor WLAN environments.

Use of this monitoring technique introduces no overhead to the network under measurement. That is, the nodes that are part of the network perform no extra work to facilitate the measurement. Additionally, no frames are injected into the network, so no bandwidth is used for the measurement aside from any used to retrieve the measurement data.

One of the main advantages of passive external capture is that a more powerful machine can be used to perform the capture. Modifications to the existing network infrastructure are not necessary. The constraints of the individual routers are not a factor because the capture node is completely separate from the network. This allows full packet capture to be performed easily. However, in the case of monitoring a wide-area network, providing a second more powerful machine at each node to perform measurement is seldom a feasible solution given both the cost and power constraints generally applicable to wireless deployments.

#### Ability to meet implementation requirements

Passive indirect capture does not meet the implementation requirements set out in section 5.1. It fails to satisfy the requirements of *scalability* and *cost* as it requires a separate measurement host for each host under measurement. This additional cost and the problems it causes for wide-spread use of passive external capture is acknowledged by Yeo *et al.* in [74].

It fails to satisfy the requirement of *accurate information capture* as it - by definition - provides a view of the channel state from the position of the monitoring host, rather than the monitored host. Due to the varying nature of the wireless channel as well as differences in the RF front-end, the monitoring host will receive frames at a different signal power and in a different noise environment than that of the intended receiver. This is significant because the monitoring host may receive frames in error when they were received by the intended receiver without error, or vice versa.

It does however meet the requirement of *limited performance impact.* Passive external capture does not impose any additional overhead on the hosts under measurement so it will not cause perceived performance degradation to end users. However, its failings exclude it as a suitable candidate for use in network-wide monitoring.

### 5.2.2 Passive Internal Capture

Passive internal capture involves instrumenting each wireless node to be measured with a "tap" that provides a copy of each frame as it is seen on the medium and does not require additional physical hardware. This direct capture at the receiver (rather than external or indirect capture at a third-party node) provides a much more accurate view of the channel state at the intended receiver and requires extra software on each node.

Passive internal capture provides many benefits in comparison to external capture. It becomes possible to see at each node how and when frames were received with an accurate view of the channel state. With this information it is possible to see how each node in range of a transmitter heard a particular frame. With each node instrumented with a direct tap, a picture can be built of the network as a whole, rather than the snapshot of a particular node provided by external capture.

Direct capture involves the use of "promiscuous" mode which turns off any hardware address filtering. Usually the Network Interface Card (NIC) will filter frames that are not intended to be processed by the host. In a measurement scenario it is generally desirable to see all frames on the medium. Disabling receive filtering increases the interrupt load on the system as well as the number of kernel- to user-space memory copy operations performed because additional frames have to be processed by the CPU.

Frames are copied to userspace using tools such as `libpcap` and `tcpdump` and then processed either online by a userspace tool or traces are transported back to an offline processing location.

#### Ability to meet implementation requirements

Passive internal capture meets the requirements of *scalability* and *limited per-host costs* as it does not require extra hardware at each host under measurement as passive external capture does.

However it fails to meet the requirement of *limited performance degradation* as it increases the load on the host under measurement. This increased load has been measured in studies such as those performed by Schulman *et al.* [57] and has been identified as a source of measurement error (failure to meet requirement *accurate information capture*) due to overloading at the measurement host causing frames to be dropped.

While passive internal capture is able to be deployed cheaply and widely, it cannot provide an accurate view of network activity and cannot operate without imposing significant performance penalties on the network. As with passive external capture, passive internal capture cannot be used for wide scale monitoring of operational networks because of these failings.

## 5.3 The WMP Kernel Measurement Framework

The Wireless Measurement Project (WMP) Kernel Measurement Framework is a new method for accurate measurement of wireless networks that meets the implementation requirements set out in section 5.1. The WMP Kernel Measurement Framework was designed and implemented as part of this thesis.

The WMP framework provides an in-kernel approach to passive internal measurement. Most existing passive internal measurement is performed by copying frames to userspace applications for analysis. The WMP framework moves the measurement task into kernelspace, reducing the overhead and increasing the accuracy of passive internal capture. This is a fundamentally different approach from the common *passive external* capture technique and provides a number of benefits over the *passive internal* capture technique.

The approach differs to existing approaches that attempt to perform passive internal measurement in kernelspace such as [59, 38] in that the measurement code is abstracted cleanly from the device driver internals. The WMP kernel measurement framework is unique in that it provides the necessary APIs for measurement tasks to be implemented without the need for researchers to understand or integrate directly with the low-level hardware device drivers. Module writers can instead focus on the implementation of the measurement task. Additionally, the WMP framework can abstract multiple underlying

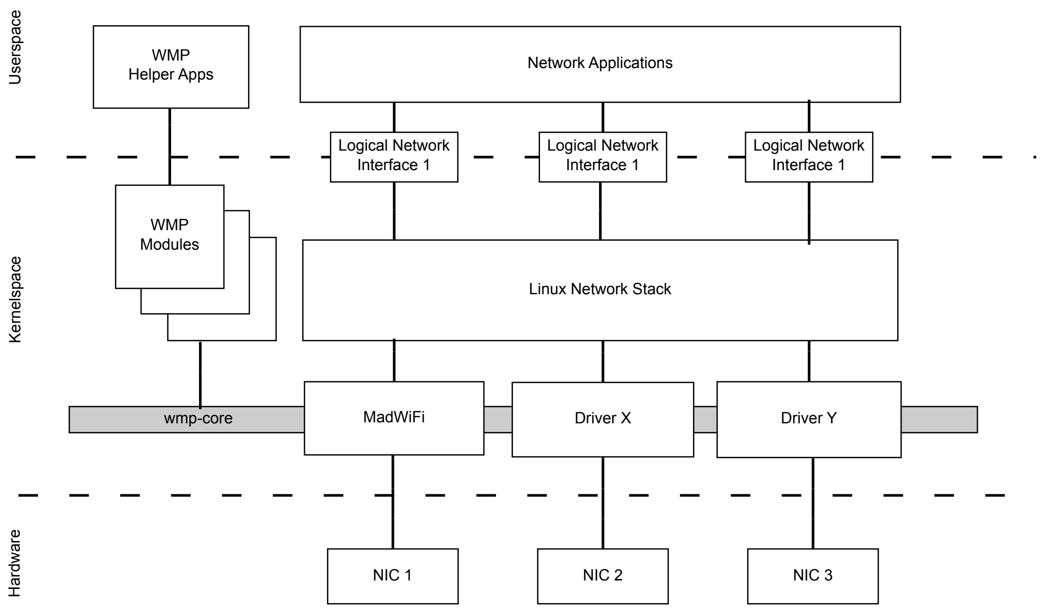

Figure 5.1: The WMP Kernel Measurement Framework Architecture

hardware devices which can reduce the amount of duplicated effort when deploying measurement tasks to heterogeneous networks.

The in-kernel API abstracts driver specific details from the implementation of measurement tasks and encourages each measurement task to be *modularised* and kept separate from the kernel device driver modules. This modular approach has three main benefits; first, the approach simplifies the implementation of measurement tasks by removing the need to integrate measurement code directly into device driver code paths. Second, the separation of measurement code from device driver code provides the ability to load and unload measurement modules independently of the device drivers, allowing measurement tasks to start and stop without affecting packet forwarding. Finally, the API allows a single measurement module to operate on any device that is supported by the WMP API, rather than requiring individual implementations of the measurement module for each device driver.

Figure 5.1 shows a high level architecture diagram of how the WMP Kernel Measurement Framework fits into the Linux kernel. The `wmp-core` layer integrates into each supported device driver and exposes the WMP API. WMP modules link against the `wmp-core` module and implement individual measurement tasks.

To investigate the hidden terminal problem, only a single device driver is instrumented with the WMP API. MadWiFi version 0.9.4 was chosen due to its wide-spread use in Linux-based wireless network deployments, and in particular its use on the Rural Link wireless network which is used for deployment testing later in this thesis. There is however no reason that the WMP API could not be applied to other device drivers and network stacks.

As well as providing the implementation framework for the measurement modules for this thesis, the WMP kernel measurement framework has been used to provide network wide long term link level statistics for a study that challenged the accuracy of traditional path loss prediction models [45].

The WMP Kernel Measurement Framework is split into two main parts; the individual Network Interface Card (NIC) device driver hooks and the measurement module API. The remainder of this section describes the framework and compares it to the passive external capture approach.

### 5.3.1 Driver Hooks

At the lowest level of the framework, modifications are made to individual wireless NIC device drivers to enable extraction of relevant metadata about each MPDU as it is processed by the driver. Figures 5.2 and 5.3 describe the metadata collected for RX and TX frames respectively. Once the metadata has been collected, the driver dispatches the frame and metadata to the WMP kernel module so that measurement modules can inspect the frame. The metadata that the WMP API currently exports is based on the metadata available from the MadWiFi driver and Atheros 5212 hardware descriptors. In future versions of the WMP API it is envisioned that specific hardware capabilities will be encoded into the metadata blocks to indicate the availability of specific metadata on individual hardware devices.

A wireless NIC generates an interrupt after a frame has been received, decoded and placed in a hardware receive queue. Frames are transferred from the device to host memory during interrupt processing. Most NIC hardware will perform receive-address filtering before the frame is placed into a receive queue to

```
struct wmp_rx_status {
        uint64_t   tsf;
        uint16_t        channel_mode;
        uint16_t        channel_frequency;
        uint8_t         padding;
        uint8_t         ieeerate;
        uint8_t         antenna;
        uint8_t         rssi_db;
        int8_t          noise_dbm;
        uint16_t        short_preamble:1;
        uint16_t        fcs_error:1;
        uint16_t        phy_error:1;
        uint16_t        has_fcs:1;
        uint16_t        rx_error:1;
};
```

Figure 5.2: RX frame metadata collected by the WMP API

```
struct wmp_tx_status {
        uint64_t        tsf;
        uint16_t        channel_mode;
        uint16_t        channel_frequency;
        uint8_t         padding;
        uint8_t         antenna;
        uint8_t         ieeerate;
        uint8_t         retries;
        uint8_t         tx_power;
        /* For devices that support multi-rate retries,
         * 'tries' counts the number of transmission attempts
         * at each rate, and 'rate' indicates the rate tried.
         */
        uint8_t         tries[4];
        uint16_t        rate[4]; /* Kbps */
        uint16_t        ack_duration[4];
        uint16_t        short_preamble:1;
        uint16_t        tx_failed:1;
};
```

Figure 5.3: TX frame metadata collected by the WMP API

reduce the number of interrupts generated and reduce the number of frames transferred from the NIC to host memory. For example, the NIC may have the ability to filter frames that are not relevant to the Basic Service Set (BSS) the station is a member of. However in many measurement situations it is desired that all frames on the medium be captured regardless of the target BSS, thus devices under measurement are generally put into promiscuous mode, disabling filtering at this level.

Multiple frames are normally transferred to the host during a single interrupt to reduce load on the host, and the NIC provides a hardware timestamp with each frame. If the timestamps were to be taken at interrupt time, they would not represent the true time that each frame was observed on the medium. This is important to measurement modules that require an accurate timestamp on individual frames.

Once the driver has frames from a physical NIC device, the frames are dispatched to the appropriate logical network devices. It is at this point, before the frames are dispatched, that the WMP kernel measurement framework takes each frame and also passes it to any measurement modules that have registered with the WMP framework for the physical device.

In the case of the MadWiFi driver, frame metadata is collected from the Atheros hardware descriptor that accompanies each frame as it is transferred from the hardware and the data is converted into the common WMP metadata format.

In the transmit path, the frames and metadata are passed to the WMP framework once the frame has finished transmission and a transmit interrupt is asserted. A frame is finished transmission either once the remote station has acknowledged successful reception, or the frame has reached its retry limit, or once the frame has been transmitted in the case of frames that do not require acknowledgement. Many drivers will transfer the frame to the wireless NIC and wait for an interrupt to indicate that the frame has been sent, hiding the process of retransmission and acknowledgement from the driver itself. Once the frame has finished transmission, the NIC will notify the driver, and the WMP framework will extract metadata about retransmissions, success or failure, etc, and make that information available to the measurement modules through the frame metadata.

### 5.3.2 Measurement Module API

Measurement modules interact with the WMP framework through a simple callback-based API. A module can register with the WMP framework to indicate its interest in frames received or transmitted by a particular physical NIC. Once a module is registered it receives a callback for each frame as it is processed by the WMP framework. This callback includes a pointer to a zero-copy reference of the raw frame data as well as the metadata collected by WMP for the frame.

The MAC header is usually discarded by the IEEE 802.11 stack when it constructs an Ethernet frame to pass to the higher layers of the network stack however the MAC header includes potentially interesting information to measurement modules, such as the retry bit and sequence number which can indicate the presence of link-level retransmissions. The zero-copy reference to each frame is a pointer to the start of the IEEE 802.11 MAC header as seen on the medium. The metadata block passed to the module includes the hardware timestamp, bitrate, antenna, and other information about the frame's reception. For example, a module may record the bitrates used by each sender by recording the bitrate from the metadata block and examining the packet contents to determine the sender address. It is assumed that modules do not attempt to modify the contents of the frame as the same copy of the frame is passed up the host network stack after all modules have finished with it. This reduces the overhead involved in copying frame data into each measurement module individually.

When a module is finished with a frame it returns from its callback. In addition to performing measurement tasks, a module can indicate whether the packet should be passed to a "monitor mode" logical network interface if one exists. Monitor mode interfaces pass a raw copy of each frame to userspace and usually include packet metadata via a Radiotap [46] or Prism II header. Monitor mode interfaces are used in passive external capture. Copying each frame into

```
int (*cb)(
  struct sk_buff *     skb,
  struct net_device * dev,
  int              direction,
  void *             metadata,
  void *             data
);
```

Figure 5.4: WMP measurement module callback prototype

userspace introduces significant overhead on resource constrained systems. In some measurement cases it may be beneficial to have a copy of some select frames passed to userspace, for example, to take a random sample of frames on the medium. A WMP kernel measurement module could perform random sampling itself and mark only a subset of frames for copying into userspace.

After the module has returned from its callback the packet continues through the host network stack as it would have previously and is passed to any monitor mode interfaces if a module has marked it. If there are no WMP modules loaded, no selective filtering of frames is performed and monitor interfaces operate as per usual.

### 5.3.3 API Details

WMP measurement modules implement a callback function (see Figure 5.4) which is called whenever a frame arrives at the WMP measurement framework. Modules register their callbacks with the framework by calling the `wmp_register_monitor_handler` function that is exported by the `wmp-core` module (see Figure 5.5). The module passes a device name, a function pointer to its callback function and an opaque data pointer that is passed back to the callback function. Once a module is registered it must be ready to start receiving callbacks immediately. Modules can unregister from the WMP framework by calling the `wmp_unregister_monitor_handler` function.

When a frame is received by the WMP framework, the callback functions of each module registered with the framework are called in the order in which they were registered. The callback function receives: a pointer to the `struct sk_buff` that contains the frame payload; a pointer to the `struct net_device` that the frame arrived on; a flag to indicate the frame direction, that is,

```
int wmp_register_monitor_handler(
  const char * devname,
  int (*cb)(struct sk_buff *, struct net_device *, int, void *, void *),
  void *data
);

int wmp_unregister_monitor_handler(
  const char *devname,
  int (*cb)(struct sk_buff *, struct net_device *, int, void *, void *)
);
```

Figure 5.5: Registration and de-registration of WMP measurement modules

whether it was received or transmitted; a pointer to the metadata block describing the frame; and the opaque data pointer that was passed to the register function.

The pointer to the metadata block must be cast to the appropriate type dependent on the direction of the frame. Frames received on an interface are described by a `struct wmp_rx_status`, while frames transmitted on an interface are described by a `struct wmp_tx_status`. Once the metadata block has been cast to the appropriate type it is up to the measurement module to perform its measurement task. Details of the data provided in each of the metadata blocks is available in the API header files.

When the callback function is finished with the frame, it must return an indication as to whether the frame should be captured into userspace via a monitor interface or simply passed to the upper network layers. It can do so by returning either `WMP_FRAME_DROP` or `WMP_FRAME_SAMPLE`.

The WMP framework does not provide any specific methods for communicating measurement results back to the user. The Linux kernel already provides several methods for kernel modules to communicate with userspace. WMP leaves it up to the module writer to decide which method best suits the measurement task. For example, the Linux kernel provides the `/proc` virtual filesystem [13] which allows kernel modules to export arbitrary data to userspace via a virtual filesystem. Alternatively, a module may implement a NETLINK socket [23] to communicate measurement events to userspace in real time.

### 5.3.4 Ability to meet implementation requirements

The WMP approach of instrumenting the wireless network stack has many advantages over passive internal and passive external capture techniques. It also meets the implementation requirements set out in section 5.1.

WMP meets the requirements of *scalability* and *limited per-host costs* as it does not require extra hardware at each host under measurement. It meets the requirement of *limited performance degradation* as it allows measurement tasks to be performed within the kernel itself, rather than relying on frames to be passed to userspace, reducing per-packet measurement overhead. Finally, it meets the requirement of *accurate information capture*; firstly because the measurement represents a more accurate view of the channel as it measures the exact same set of frames as the host under measurement (when compared to passive external capture); and secondly because the reduced per-packet measurement overhead leads to the measurement system suffering from less packet loss (when compared to passive internal capture).

Passive external capture has some advantages over the WMP approach. It is a very simple method for quickly measuring a small area, such as when performing site-surveys or access point coverage checks. If the measurement task does not require the accurate capture of all frames on the medium, or if the hardware that is used for the capture point is sufficiently powerful enough, or if the measurement task only requires very few measurement points, then deploying passive external capture is very simple. Additionally, passive external capture is separate from the host under measurement, so it does not pose any additional overhead on the host under measurement.

Passive internal capture is conceptually simpler than the WMP approach, and significantly easier to deploy as it does not require the creation of kernel modules to implement the measurement.

The following sections discuss the practical details of implementing the two methods for detection of hidden terminals within the WMP kernel measurement framework.

## 5.4 Implementation of the network graph method

This section discusses the implementation of the network connectivity method for detecting hidden terminals that was introduced in section 3.1 . The network connectivity method uses information from hosts in the network collecting data about the hosts they can receive packets from. This data is then collated at a central point and a connectivity graph is constructed. Hidden terminals can be found by applying graph theory.

The process is split into three parts; the raw data collection that must be performed per-host, the transfer of raw data to the central processing point, and the processing of the data at the central collection point. Each of the parts are discussed separately.

### 5.4.1 Raw data collection

The raw data collection phase is performed by a WMP kernel measurement module on each host in the network. The wireless NIC is put into promiscuous mode so that all frames that are able to be decoded are passed to the host, regardless of their destination. As frames are received by the WMP measurement module the sender addresses are extracted. In this way, the module builds a list of all terminals that the host under measurement can hear, regardless of whether the terminals are part of the same BSS or not. The module maintains a list of sender addresses it has seen and exports the list via the `/proc` virtual filesystem.

This process has very little overhead and does not require measurement traffic to be injected into the network. However, when a frame is received that fails its FCS (as indicated by the `rx_error` flag) the frame contents cannot be trusted and hence the sender address cannot be extracted. Frames may fail their FCS for a number of reasons, for example simple corruption due to fading effects, poor receive SNR due to path attenuation, or frame truncation due to collisions or desynchronisation.

In the case of corruption due to fading or truncation due to collisions, it is likely that a frame from the same sender will be heard correctly within a relatively short time period. In the case of high path attenuation, it may be the case that the time period before receiving another frame without error is relatively high. In this case, the sender is contributing to the carrier-sense behaviour of the host, but it is unable to have its transmissions reliably decoded - and hence the source of the transmissions is unable to be determined.

Active measurement can be used to try and reduce the period taken to positively identify senders in the network. By periodically injecting frames with a known payload into the network and recording reception of such frames, a graph can be created which shows with a high level of confidence the connections between terminals. These active measurement frames contain a forward error corrected payload which protects the sender address of the frame, allowing frames to be identified more reliably when received in error. This improves the ability for terminals to be identified when they are on the edge of receive range.

A simple daemon process periodically injects 802.11 frames through a monitor mode interface, which are then transmitted by the wireless NIC. These frames are addressed to the broadcast address, so they are processed by all listening terminals. The 802.11 MAC header is intentionally obfuscated by setting the frame control bits to 1's (an invalid combination) so that the driver does not pass the frame up the network stack of any host it is received by. The frames are however processed by the WMP kernel module(s).

The collected data is exported by the WMP measurement module via the `/proc` virtual file system. Raw packet counts are kept for each terminal detected, as well as information such as which BSS the terminal belongs to, the terminal's average signal-to-noise ratio, and the channel the BSS is operating on.

### 5.4.2 Data Processing

A small daemon that runs on each host periodically collects and compresses the data exported by the WMP measurement module. The compressed data is then sent via an HTTP POST to a central monitoring server. In the event of a failure to send the measurement data, the daemon will retry a number of times before giving up.

At the central processing server, a web CGI uncompresses the results as they are POSTed by each monitored host and stores the data in a local database. The database contains historical connectivity data for the entire network under measurement.

With network connectivity data being collected from each host, hidden terminals can be found by generating connectivity graphs and applying graph theory. Each network interface is represented by a vertex in the graph, and directed edges are drawn between vertexes when there is a record of packet reception. New vertices are added for each transmitting interface seen, so all terminals appear on the graph, whether they are under measurement or not. For example, if terminal A has recorded receiving a frame from terminal B, a directed edge will be drawn from B to A, even if terminal B is not under measurement.

During construction of the graph, vertices that are added to the graph that are not under measurement are marked as "external nodes". An external node represents a terminal which was not under measurement, but was detected by a terminal that was under measurement. These external nodes have no incoming edges since they do not report any connectivity information, so no assumptions can be made about their part in hidden terminal activity. However, while the method cannot draw any conclusions about external nodes' participation in hidden terminal activity, the presence of such terminals on the graph may help to give information about the density of wireless activity in the area, and hence the possibility of hidden terminal activity.

## 5.5 Implementation of the timing analysis method

The timing analysis method described in section 3.2 detects violations of the IEEE 802.11 DCF by examining the inter-frame spacing of consecutive packets. Violations of the DCF are likely to indicate the presence of hidden terminals, and the rate at which these violations are occurring can be used to estimate the total probability of collisions due to hidden terminals.

The timing analysis method can be run on a single wireless network interface to measure the rate of hidden terminal events at a single point in the network. Alternatively it can be run on every wireless network interface in the network as part of a long term measurement system. This is different from the network connectivity method, which *requires* connectivity information from as many interfaces in the network as possible.

As with the network connectivity method, the implementation of the timing analysis method is split into two parts; raw data collection on the measured hosts and processing on a central server.

### 5.5.1 Data collection

The timing analysis method is implemented as a WMP kernel module. Observing the properties of inter-frame spacing requires operating on consecutive pairs of received frames. The WMP module retains information about the previously received frame and compares it to the frame that is currently being processed.

When a packet is processed the timestamp at the start of reception, $ts_{start}$, is calculated using the method described in section 4.2. The inter-frame space is determined by subtracting the timestamp at the end of reception, $ts_{end}$, of the previous packet from the $ts_{start}$ of the current packet. If the resulting inter-frame space is smaller than a configurable threshold—the default case is $16\mu s$—a *packet pair record* (Figure 5.6) is created. The packet pair record describes the packet pair involved in the DCF violation and includes start and end timestamps, bitrates, signal levels, sender addresses if applicable, flags to indicate whether the packets were received in error and the length of the received packet in octets.

Heuristics are applied to packet pairs as they are processed to improve the DCF violation detection accuracy and are described in the following sections. First, frames are tested for truncation to avoid mis-classification of collisions as

```
struct ppr_t {
        /* Timestamp when the difference was measured */
        struct timespec tv;

        int64_t difference;

        unsigned int    last_octets;
        uint64_t        last_start;
        uint64_t        last_end;
        unsigned int    last_ratecode;
        unsigned int    last_rxerror;
        uint8_t         last_sa[19];
        unsigned int    last_type;
        unsigned int    last_subtype;
        unsigned int    last_truncated;

        unsigned int    octets;
        uint64_t        start;
        uint64_t        end;
        unsigned int    ratecode;
        unsigned int    rxerror;
        uint8_t         sa[19];
        unsigned int    type;
        unsigned int    subtype;
        unsigned int    truncated;

        int             shpreamble;

        struct ppr_t *  next;
};
```

Figure 5.6: A Packet Pair Record describing the packet pair involved in a DCF violation. The Packet Pair Records are stored as a simple linked list.

SIFS violations. Secondly, the length of the inter-frame space must be checked to ensure that the correct PLCP preamble length is being used and adjusted if not.

### 5.5.2 Detection of truncated frames

A truncated frame followed very closely by another frame is usually a symptom of a collision. While collisions are in general caused by hidden terminals, they are not used by the timing method as it is searching for SIFS violations rather than collisions. The difference is that SIFS violations do not cause frame overlap, rather they cause frames to arrive "too soon" after the previous frame.

Truncated frames cause the timing analysis method to erroneously classify the collisions that caused them as SIFS violations. Truncated frames therefore must be detected and marked as such so that the timing method can discard these frame pairs.

If truncated frames are not removed from the sample set, a large number of 0, 1 and 2 $\mu s$ inter-frame spacings are recorded and are considered by the method as SIFS violations when they are in fact not. It is not enough to simply remove all instances of 1 and 2 $\mu s$ frame spaces either, as these are valid values for non-colliding inter-frame spacings in the presence of hidden terminals. Instead, the first frame must be inspected to see if there are any indications that the frame was truncated. If so, the frame pair is not considered as causing a SIFS violation and is instead considered as a collision.

Detecting truncated IEEE 802.11 MPDUs is not straight forward. The PLCP header includes a length field, but this is not generally made available to the host operating system. The 802.11 MAC header that is available to the host contains no length field. A heuristic must therefore be used in order to determine if a frame is truncated.

Frames that pass their FCS are immediately classified as not truncated. The remaining frames are checked against several rules to make a best-effort guess at their truncation. The frame length is first checked against the minimum IEEE 802.11 MPDU length. Then, if the ethertype of the MPDU payload is recognisable the length of the frame is compared to expected frame lengths for the payload. If the ethertype is not recognisable, the heuristic returns "unknown".

The ethertypes supported are: LLC/SNAP, ARP, IP, PPPoE, and EAPOL. Care is taken to ensure that hardware data padding bytes are taken into account, as well as variable length 802.11 MAC headers such as optional QoS and four-address frame formats. If these payloads can be decoded far enough that a length field is found it is compared to the length of the received payload. If the received payload is shorter than the expected payload length, then the heuristic returns "truncated".

In this way, a simple best-effort guess can be made to determine if frame truncation has occurred and the mis-classification of collisions as SIFS violations can be reduced.

### 5.5.3 Detection of preamble length

The methodology requires the total length of the PPDU to be known in order to accurately determine the start timestamp of the PPDU. The length of the PLCP preamble varies depending on the PHY used by the transmitter and other options of the transmitting terminal and so the total length of the PPDU is not easily calculated. These options are not known by the receiver - there is no indication that is given to a receiving host about the PLCP preamble type that was used by the transmitter. In the absence of concrete information regarding the PLCP preamble length, a heuristic must be applied. This heuristic relies on the prior removal of frames that are known to be truncated due to collisions which was presented in the previous section.

The heuristic is as follows: PPDUs sent with a PSDU encoding rate of 1Mbit/s are sent with a long PLCP preamble, as the IEEE 802.11 standard requires this to ensure backward compatibility amongst stations. All other PPDUs have their PLCP preamble length set via an option on the transmitter. The timing method first *assumes* that a long preamble was used. The total PPDU length is calculated under this assumption and the inter-frame spacing is then calculated. If this assumption yields the impossible result of an inter-frame spacing with a large negative magnitude then assume a short preamble was used.

Calculation of PPDU length is further complicated by the multiple High Rate PHYs available. For example, a terminal operating in 802.11g mode has the option of using either OFDM or DSSS encoding schemes for the PLCP header, depending on the capabilities of the BSS. MPDU fields only indicate the capabilities of the stations, not the actual use of these optional parameters, and again, information about the PPDU decoding is not passed to the host so it is difficult to determine which encoding and hence which *TXTIME* calculation to use. As such, the implementation has been restricted to the IEEE 802.11b PHY as it requires less heuristics to determine frame length and still demonstrates the viability of the method without loss of generalisation. The implementation of an IEEE 802.11g PHY capable module is left as future work.

### 5.5.4 Identifying senders

Finally, to resolve hidden terminal collision based performance problems it is very useful for a network operator to know how hidden terminal collisions are distributed among the remote stations. In order to identify the terminals that are involved in a particular hidden terminal collision, it is necessary to identify the sender address of each packet in the pair.

The sender address of a packet is not always readily available. IEEE 802.11 Control frames, such as ACK and CTS frames do not contain a sender address. Frames which are truncated, such as those involved in a collision, may not contain enough of the payload to contain a sender address. Even though the timing method discards packet pairs in which the first frame is truncated, the second frame may be truncated due to a subsequent collision. Finally, frames which fail their FCS may contain the sender address but it may be in error. As such, only SIFS violation events in which both frames are able to be correctly decoded are considered for sender identification. Packet pairs in which one or both frames are damaged are counted towards the overall rate but are not attributed to individual sender pairs.

### 5.5.5 Data output

The module records every packet pair that has an inter-frame space below the defined threshold. These packet pair records are then exported via a CSV formatted file in the `/proc` virtual file system. One file is created per interface under measurement. When this file is read by a userspace helper process the record list maintained in the kernel module is cleared to limit the maximum memory use of the module. The number of records that the kernel module will retain is limited and once this limit is reached, old records are dropped from the tail of the list.

The file is prepended with a header containing metadata about the data collection process. This includes the number of packets processed by the kernel module, the number of events detected, the number of event records dropped due to memory constraints, the device's hardware address and its interface name and the total channel time occupied by frames received by the interface.

After the header, each record is printed separated by line endings. The data in each CSV record consists of: the inter-frame space in microseconds and then for each packet in the pair, the MPDU length in octets; the $ts_{start}$ and $ts_{end}$ timestamps of the PPDU; the PSDU bitrate; the error flag; the sender address if available; the frame type and subtype; and the results of the truncation and preamble heuristics.

### 5.5.6 Post processing

A small userspace helper process on each measured host periodically reads the contents of the /proc file and transmits its contents back to a central server via an HTTP POST. The POST handler on the server inserts records into an SQLite database for future processing.

For IEEE 802.11b PHYs (HR/DSSS), if the inter-frame space $x$ is $0\mu s < x < 8\mu s$, and the first frame of the packet pair was not truncated, the event is counted as a violation of the SIFS. The sender of the second frame caused a violation as it was unable to detect the transmission of the sender of the first frame. The sender address of each frame may be available in the event record if the frames contained sender addresses, which allows the event to be counted against the sender of the second frame.

The total number of frames received by the interface and the total channel time occupied are available as part of the update and along with the counts of SIFS violations, the total number of hidden terminal collisions can be estimated by using the equations presented in 3.2.

## 5.6 Chapter Summary

This chapter has presented a set of implementation requirements which constrain the implementation of the hidden terminal detection methods in order to make them widely usable by network operators. It has discussed existing techniques for wireless measurement and has shown that existing techniques do not meet these requirements.

This chapter has introduced the WMP kernel measurement framework for low-overhead, accurate measurement of wireless networks. WMP instruments the wireless stack directly in the kernel, giving measurement modules access to exactly the same frames as were processed by the host under measurement while not imposing extra overheads usually introduced by passive internal capture. The WMP approach meets the implementation requirements.

Finally, each of the hidden terminal detection methods were implemented within the WMP framework and the details of each implementation were discussed. The following chapter presented results from a real-world deployment of the implementations discussed in this chapter. It shows that they can be used to detect and measure the effect of hidden terminals in a real, operational wireless network.

# Chapter 6

# Deployment Results

This chapter shows that the combination of network connectivity and timing methods can be used on an operational network to detect the potential for, and measure the effect of, hidden terminals. It also demonstrates how this information can form the basis for tools that are useful to network operators to help diagnose real world network problems.

The methods described in Chapter 3 were validated in a laboratory setting in Chapter 4 and their implementation within a wireless measurement framework for Linux-based wireless routers was described in Chapter 5. This chapter presents results from case studies based on deployment of the hidden terminal detection methods within an operational, commercial IEEE 802.11 network.

The methods were implemented within the WMP Kernel Measurement Framework (Chapter 5) and were deployed across 85 wireless routers comprising the backbone and access network of a commercial rural wireless ISP in Hamilton, New Zealand.

The Rural Link no.8wireless network is primarily made up of 5.8GHz point-to point backbone links and a 2.4GHz point-to-multipoint access network, based on IEEE 802.11 technology. Client CPEs and the backbone routers run a customised GNU/Linux operating system and use Atheros 802.11 chipsets, making the network a suitable target for measurement using the WMP Kernel Measurement Framework described previously.

All of the wireless interfaces in the network were measured. Three cases are described in detail in this chapter to illustrate the various ways in which the methods can give network operators useful information about the impact hidden terminals may be having on the performance of their network. Each case study is introduced with a description of the segment of the network and its traffic characteristics followed by results of the connectivity and timing methods and a discussion of the results obtained.

The connectivity graphs (e.g. Figure 6.2) show hidden terminals from the perspective of the interface under measurement by showing the connectedness of terminals that are heard by the interface under measurement. The connectivity graphs shown do not include external terminals that were not under measurement as connectivity data for such terminals is unavailable. Such external terminals may however cause hidden terminal collisions at the interface under measurement. These will be detected by the timing method.

The remote terminals shown in the connectivity graphs may hear other terminals that the interface under measurement cannot. In this case, the terminals are not considered hidden terminals with respect to the interface under measurement as they cannot produce a collision with it. Such terminals are generally referred to as "exposed" terminals [11], as they prevent the affected terminal from transmitting to a third terminal that would otherwise not cause a collision due to their participation in CSMA. Exposed terminals are not the focus of this thesis and hence are not included in the connectivity graphs, however they can be detected by the network connectivity method.

The interfaces to study were chosen from the set of measured interfaces by measuring the connectivity of each interface, ordering them, and choosing three examples that were indicative of different levels of connectivity. These three examples were then studied in detail and their results are presented in the following sections. The interfaces were not chosen based on their measured collision rates or any other factor that made them more or less likely to show "interesting" results. In this way, three interfaces were chosen which illustrate three different hidden terminal scenarios without bias toward any particular result.

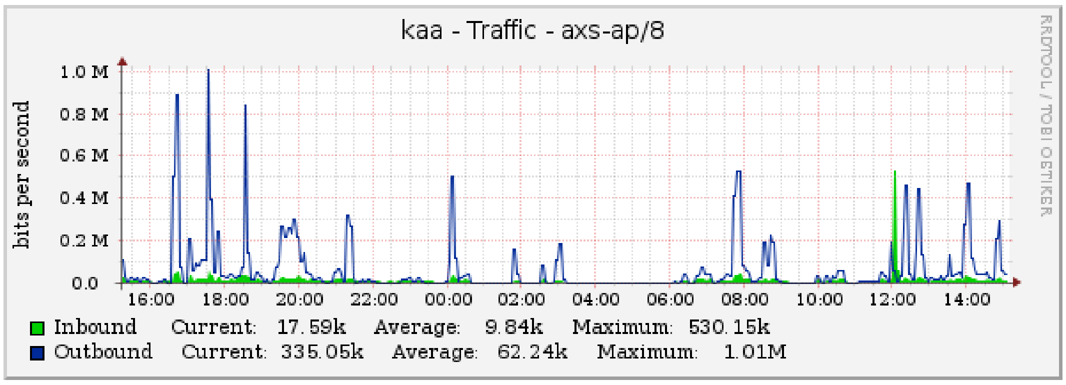

Figure 6.1: Indicative traffic profile of the KAA access point interface over a 24hr period with 5 minute averages. This graph shows traffic generated by directly connected clients. Any 802.11 traffic that can be heard by this interface but is filtered (i.e. not part of the BSS) will not be included in this graph.

## 6.1 Case Study 1

The first case study shows a site with good connectivity between its clients. The "KAA" repeater site in Whakamaru, New Zealand is an edge node of the 5.8GHz wireless backbone network. The site itself is in a remote area which is an otherwise quiet radio environment at 2.4 GHz. It hosts a 2.4GHz access point interface that services nine client CPEs. On an average day, the clients connected to the site generate an average of 100 Kbps of aggregate traffic, peaking to approximately 1 Mbps (Figure 6.1).

Seven of the nine clients are within a 300 meter radius of the site. The other two clients are approximately 1.5 kilometres from the site. Network connectivity (Figure 6.2) shows that the clients are mostly well connected which is expected in cases where clients are within such close proximity of one another. Such well-connected clients are unlikely to exhibit significant numbers of hidden terminal collisions.

The timing method shows a very low level of detected SIFS violations, and hence estimates a low hidden terminal collision probability (Figure 6.3).

This case shows that, for a set of well connected sites, very few hidden terminals collisions are detected by the proposed methodologies, as expected. This information would allow network operators to be confident that hidden terminals are not causing performance problems. Performance problems in a scenario like this one would more likely be attributed to other factors such as

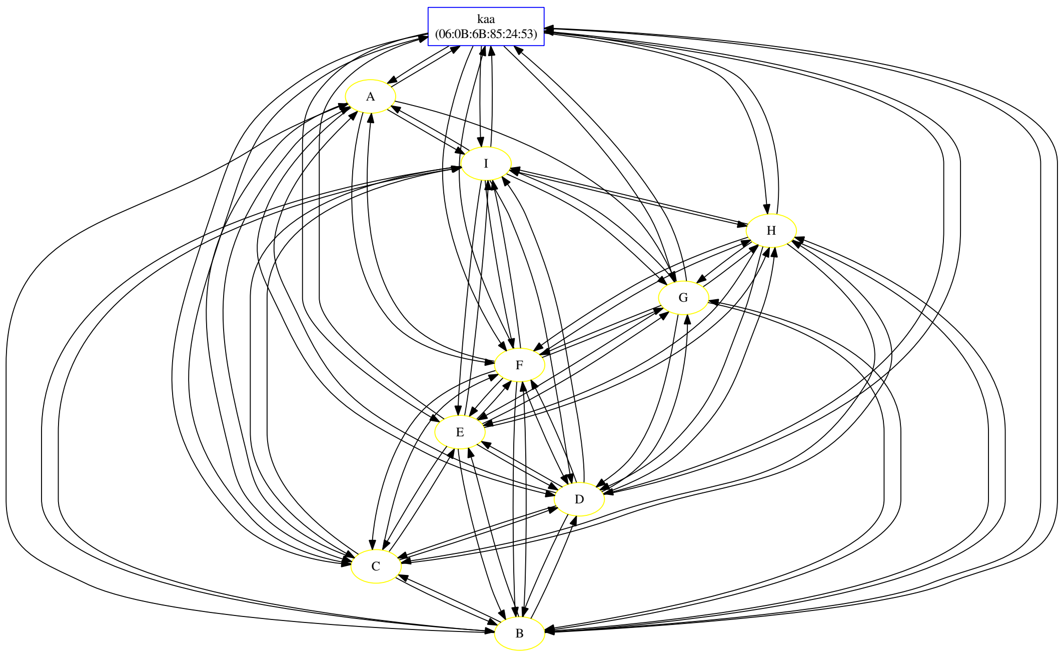

Figure 6.2: Connectivity graph for KAA site. The blue square node indicates the access point interface under measurement. The yellow circular nodes indicate terminals that the interface under measurement can hear. A directed edge indicates connectivity from the tail to the head node.

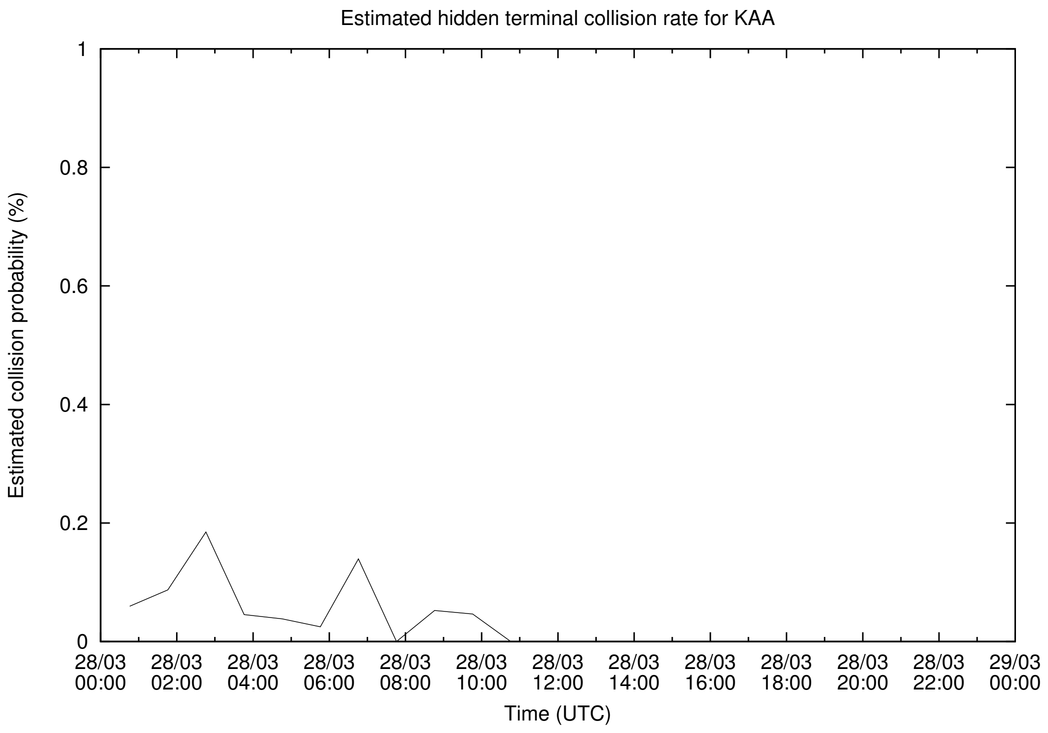

Figure 6.3: Estimated hidden terminal collision rate based on timing method at KAA site.

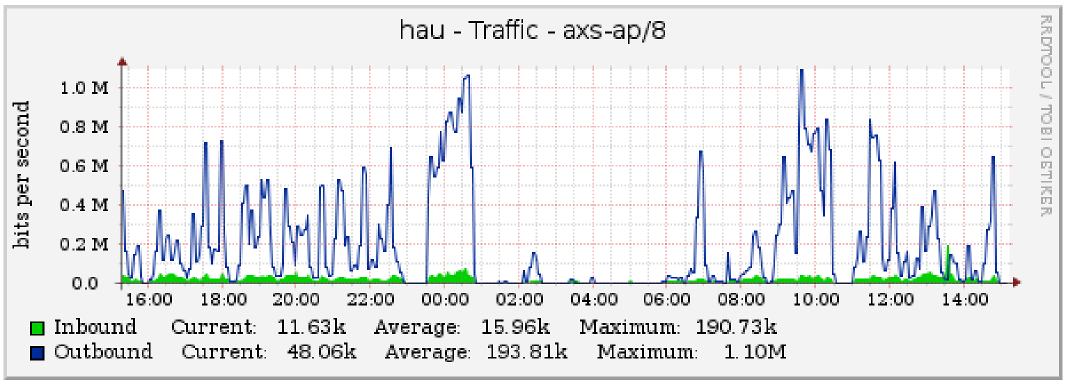

Figure 6.4: Indicative traffic profile of the HAU access point interface over a 24hr period with 5 minute averages.

non-802.11 interference, signal fading, hardware faults, or other effects.

## 6.2 Case Study 2

In this case study, an access point on the network is shown to have poor connectivity between its clients which would suggest a high possibility for hidden terminal collisions. However, the timing method shows few hidden terminal collisions actually occurring during the period of measurement.

The "HAU" repeater site in Ngahinapouri, New Zealand, has eight clients and the network connectivity graph shows relatively poor connectivity between them. In particular, several of the clients can only hear one or two of the other clients, suggesting a high probability of hidden terminal collisions.

The timing method shows that hidden terminal collisions are detected and estimates that they cause approximately 1% of packets to collide. The directly connected clients that are causing the majority of identified SIFS violations as shown in Table 6.1 are the clients that are most poorly connected in the network connectivity graph (Figure 6.5). The low hidden terminal collision rate is consistent with the traffic patterns of the clients involved. On an average day, the clients generate an average of 200Kbps of aggregate traffic, peaking to 1.1 Mbps. If the clients are all relatively light users of the network fewer collisions are likely to occur even when connectivity shows a high potential for hidden terminal collisions. The indicative traffic graph (Figure 6.4) shows traffic only from directly connected clients, however the estimated hidden terminal

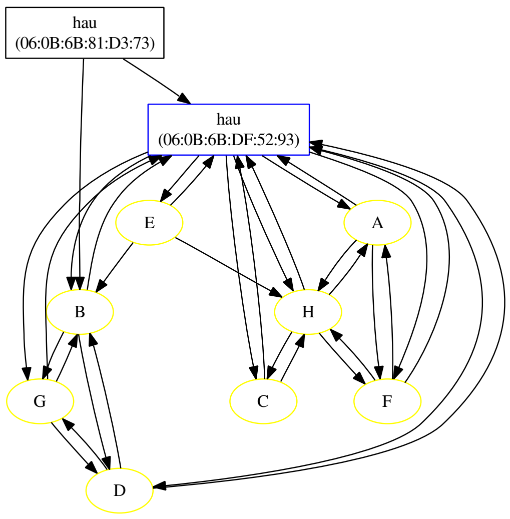

Figure 6.5: Connectivity graph for HAU site. Note that while there are no non-directly connected clients visible, the directly-connected clients form two distinct groups that are hidden from one another.

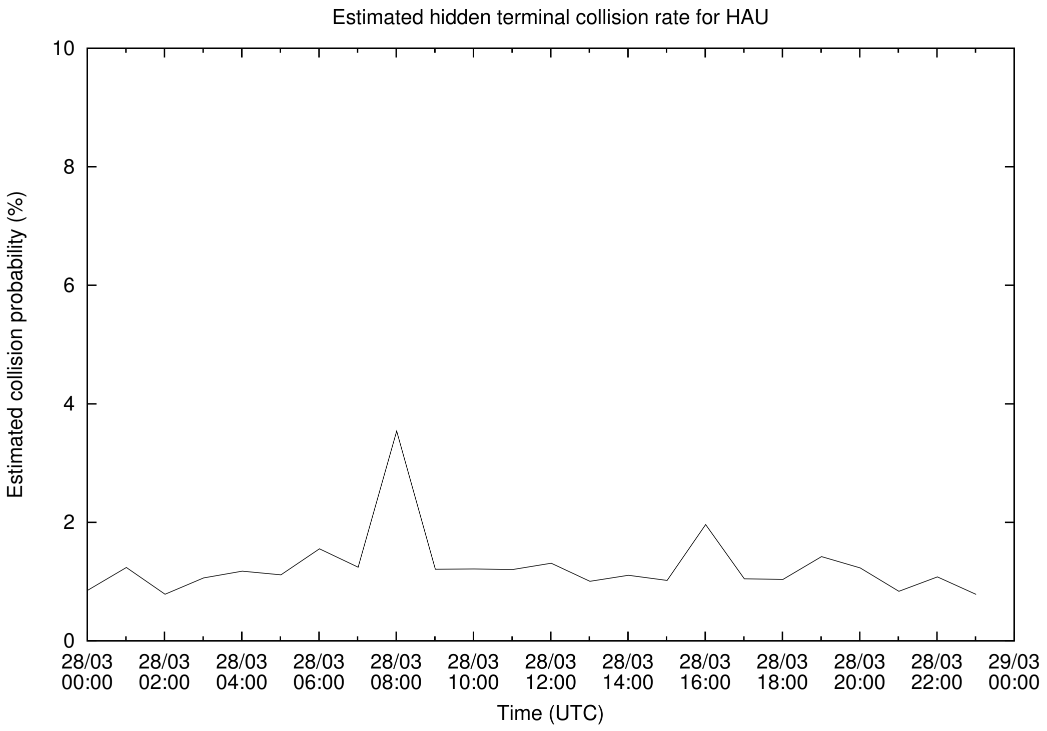

Figure 6.6: Estimated hidden terminal collision rate based on timing method at HAU site

Table 6.1: Top identified sources of SIFS violations at HAU over a 24 hour period.

| Sender MAC Address | SIFS Violations Detected | Directly Connected |
|---|---|---|
| 00:18:0a:01:5d:04 (G) | 518 | Yes |
| 00:11:3b:11:6c:69 | 377 | No |
| 00:15:6d:53:ee:60 | 298 | No |
| 00:18:0a:01:5a:43 (B) | 162 | Yes |
| 00:16:e3:64:8e:70 | 150 | No |
| 00:12:cf:81:fe:bd (E) | 39 | Yes |
| 00:12:cf:cb:93:0 (C) | 18 | Yes |
| 70:72:cf:17:2b:a1 (H) | 13 | Yes |
| 00:18:0a:01:5c:da (D) | 10 | Yes |
| 70:72:cf:17:2d:b (A) | 6 | Yes |
| 00:12:cf:80:69:37 (F) | 5 | Yes |

collision rate is no smaller than 0.8% throughout the 24 hour period. Table 6.1 shows a large number of SIFS violations are detected from non directly connected clients, which could explain this background level of collisions, even when traffic from directly connected clients is low.

This case study shows the different strengths of the two methodologies. The network connectivity method finds the *potential* for hidden terminal collisions while the timing method measures the current *impact* of hidden terminal collisions. Network operators can use the information from the network connectivity method to identify areas in the network that require monitoring. If the traffic in this part of the network increases it is likely that the number of hidden terminal collisions would also increase.

## 6.3 Case Study 3

The final case study shows a site that has very poor connectivity between its clients and very high hidden terminal collision detections. The "PKU" repeater site in Te Awamutu, New Zealand is situated on an elevation with wide ranging views of the Waikato plains. It is also an edge node on the 5.8GHz wireless backbone, with a 2.4GHz sector antenna for client access.

The site itself has 11 directly connected clients, however the network connectivity method shows it is able to receive transmissions from 22 transmitters in total. The non-directly connected transmitters are part of the same net-

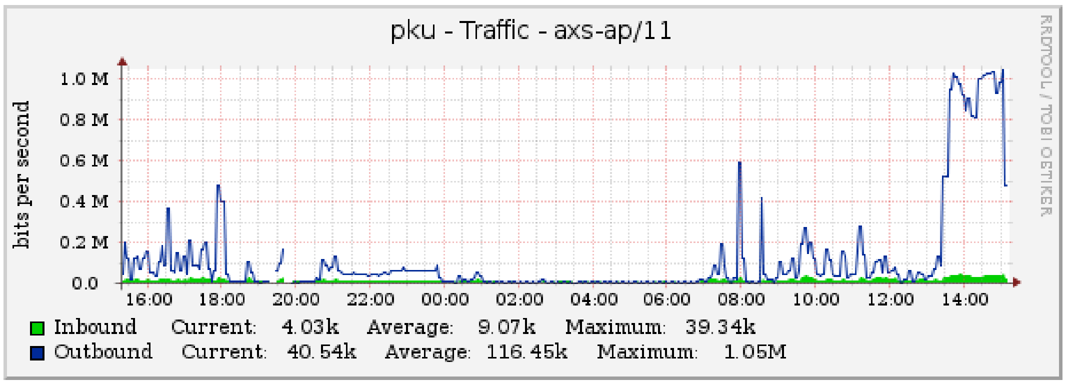

Figure 6.7: Indicative traffic profile of the PKU access point interface over a 24hr period with 5 minute averages.

work under measurement. The directly connected clients are poorly connected amongst themselves due to long link distances and the additional transmitters that can be heard by the access point interface add to the potential for hidden terminal problems. In this case, Figure 6.8 shows a very high potential for collisions to be caused by hidden terminals.

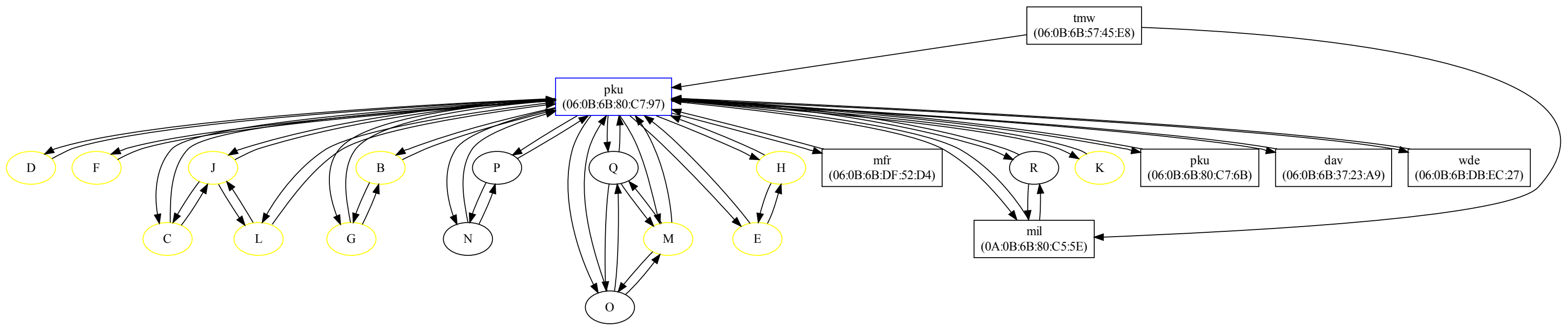

Figure 6.8: Connectivity graph for PKU site. Note that several other site interfaces can be heard by the PKU interface under measurement, including another interface on the same host. These non-directly connected interfaces are indicated in black. Most of the client CPEs are poorly connected to each other, and there are several client CPEs visible that are not directly connected to the PKU site.

The timing method confirms this theory (Figure 6.9), estimating between a 5 and 13% collision rate due to hidden terminals. The collision rates are consistent over time, showing that this segment of the network is badly affected by the hidden terminal problem.

Table 6.2 shows the remote terminals that the methodology identified as causing collisions and the number of detected SIFS violations caused by each. The table shows that the non-directly connected terminals were the main source of SIFS violations. Further investigation into the problem showed that the non-directly connected terminals were transmitting on the same channel as the terminal in question.

Although the no. 8wireless network operators were aware that the interfaces involved were operating on the same channel, it was thought that the sites were sufficiently separated geographically that transmissions would not interfere. With this new information in mind, the channel allocation scheme was modified to reduce channel overlap. Figure 6.10 shows the connectivity after the change. The number of non-directly connected terminals decreased from 11 to 2. The estimated rate of hidden terminal collisions (Figure 6.11), decreased from approximately 10% to 2%. The lead to an overall improvement in the performance of the network.

This case study shows that the combination of the network connectivity and timing methods can provide relevant information to network operators that allows them to discover, diagnose and resolve hidden terminal based network problems. It demonstrates that these problems may cause a significant degradation of network performance but are not always visible to operators. The use of WMP found this problem when passive measurement would not have, in part because WMP could be deployed on all terminals in the network without significant cost. The connectivity method highlighted the potential for hidden terminal collisions to occur and the timing method showed the extent of those collisions. It highlighted that non-directly connected terminals were the largest contributors. This in turn allowed the network operators to make configuration changes that improved the performance of the network by reducing the number of hidden terminal collisions.

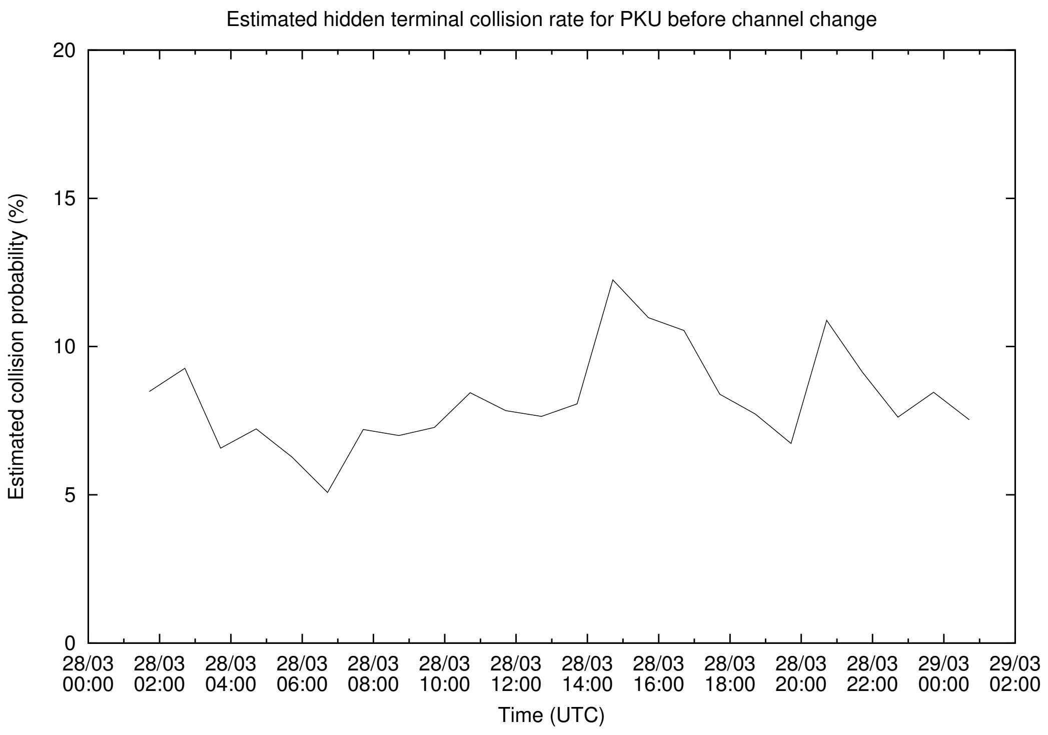

Figure 6.9: Estimated hidden terminal collision rate based on timing method at PKU site

Table 6.2: Top identified sources of SIFS violations at PKU over a 24 hour period. This table shows the majority of SIFS violations were caused by non-directly connected interfaces, and in particular, other access point interfaces on the network.

| Sender MAC Address | SIFS Violations Detected | Directly Connected |
|---|---|---|
| 00:0b:6b:80:c5:5e (mil) | 570 | No |
| 70:72:cf:17:26:fd | 362 | Yes |
| 00:0b:6b:57:45:e8 (tmw) | 318 | No |
| 00:0b:6b:db:ec:27 (wde) | 269 | No |
| 06:0b:6b:80:c5:5e (mil) | 154 | No |
| 00:0b:6b:37:23:a9 (dav) | 96 | No |
| 00:12:cf:d3:36:03 | 62 | Yes |
| 00:12:cf:c9:3e:f8 | 53 | Yes |
| 00:18:0a:01:5c:81 | 38 | Yes |
| 00:12:cf:83:82:15 | 35 | Yes |
| 00:0b:6b:84:b4:5a | 23 | Yes |
| 00:12:cf:81:80:5d | 20 | Yes |
| 00:0b:6b:84:b5:54 | 15 | Yes |
| 00:12:cf:d3:35:17 | 13 | Yes |
| 00:12:cf:d3:9b:49 | 8 | Yes |
| 00:0b:6b:0a:83:ef | 8 | No |
| 00:12:cf:d3:35:21 | 8 | Yes |

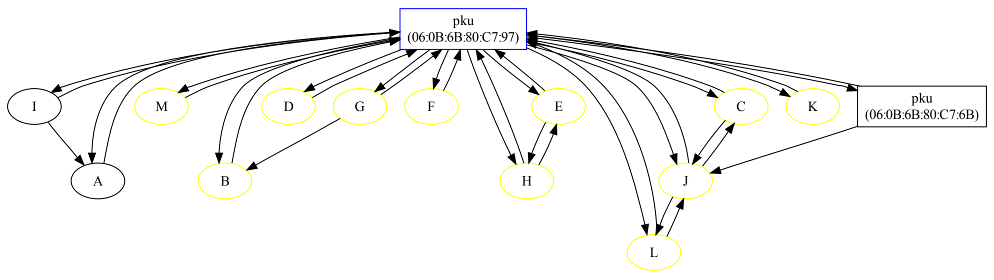

Figure 6.10: Connectivity graph at PKU after channel change. Only two non-directly connected terminals are now present in the graph. The connectivity between directly connected client CPEs remains similar, as expected. Hidden terminal collisions are still likely due to poor connectivity between directly connected visible terminals as well as the removal of other site interfaces should reduce the overall collision rate.

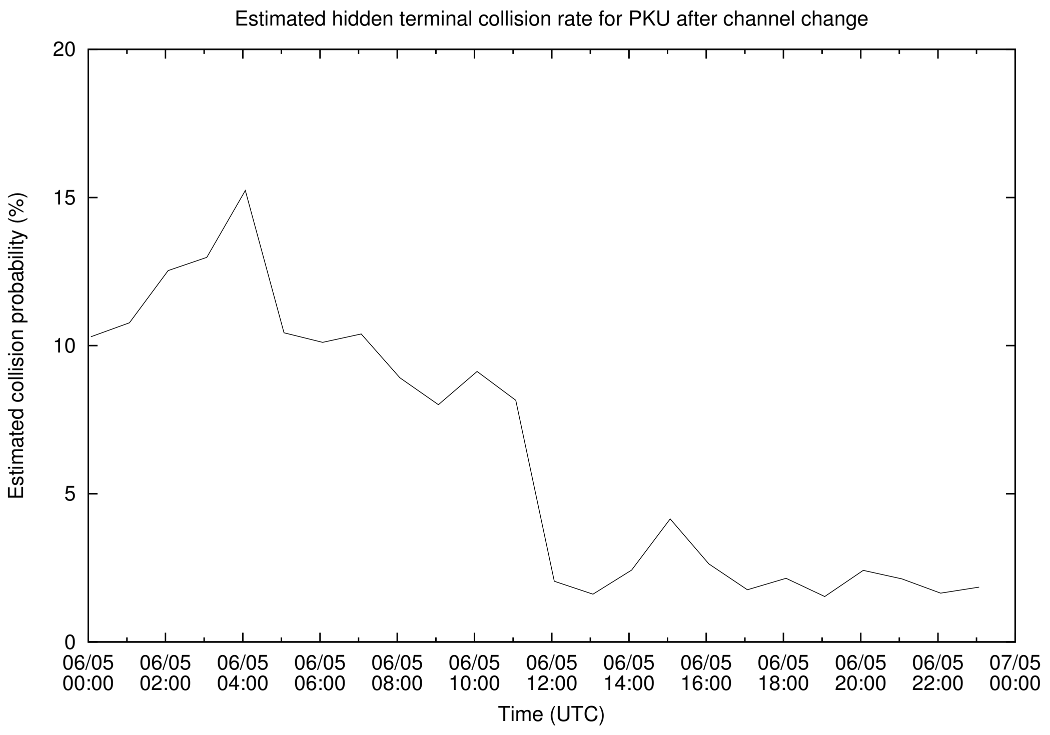

Figure 6.11: Estimated hidden terminal collision rate based on timing method at PKU site. At midnight local time (12 noon UTC), the channel was changed based on previous connectivity data. The drop in hidden terminal collisions is reflected here.

## 6.4 Chapter Summary

This chapter has presented three case studies in which the network connectivity and timing methods were used to characterise hidden terminal activity at individual wireless repeater sites on an operational commercial wireless network.

In all three cases, the methods provided practically useful information to the network operators. In one case, the methods diagnosed a significant problem and the network operators were able to make changes to the network which improved overall performance. Without the use of the hidden terminal detection methods, it is unlikely that the channel planning issue would have been detected.

The information returned by the hidden terminal detection methods was analysed manually for the purposes of this chapter. However, the analysis is easily automated, allowing for integration into automated network monitoring systems and analysis by non-expert operators.

This chapter has shown that the hidden terminal detection methods presented in this thesis are practical and useful in a real world, operational wireless network environment. It has also shown empirically that the effect hidden terminals can have on a network is significant, further confirming existing literature.

# Chapter 7
# Conclusions and Future Work

## 7.1 Thesis Summary

The hidden terminal problem is an important issue in wireless networks based on the CSMA medium access control scheme. In particular, tools are required to reduce the difficulty of diagnosing complex issues such as the hidden terminal problem in community networks operated by non-experts. In this thesis, new methods are presented for the detection and measurement of the hidden terminal problem in networks based on commodity hardware and software platforms.

A method that measures network connectivity can be used to detect areas of a network in which hidden terminals have the *potential* to affect the performance of the network. A complimentary method that uses the strict timing requirements of the IEEE 802.11 Distributed Coordination Function can be used to detect *actual instances* of the hidden terminal problem. An extension of the timing method allows for an estimation of the *overall impact* hidden terminals are having on the performance of the network by estimating the total rate of collisions caused by hidden terminals.

The methods are validated in a controlled laboratory environment through a set of experiments in which known rates of hidden terminal collisions are generated. It is shown that the overall rate of collisions caused by hidden terminals can be estimated based on detection of SIFS violations during times of low to medium packet loss. During times of high packet loss, the estimation accuracy is reduced.

The methods are implemented within a new framework for instrumenting Linux based wireless routers. This framework allows for measurement to be performed on wireless routers without significant impact on the performance of the network. The in-kernel approach to measurement provides an abstracted API against which measurement tasks can be implemented without the need for module writers to integrate directly with individual device drivers. This approach simplifies the implementation of measurement tasks and provides a number of benefits over existing techniques for performing wireless measurement. The performance penalty of performing measurement in userspace is reduced when compared to passive internal capture. The need for external measurement hardware is removed and the overall accuracy of measurement is improved when compared to passive external capture.

Three case studies are presented in which the hidden terminal detection methods are deployed in an operational rural wireless network. In all three cases, the methods are able to provide useful information to the network operators about the state of hidden terminals in the network. In one case, the methods highlight an area of the network which is suffering from hidden terminal problems and identifies the source of the hidden terminal collisions. The network operators were able to make a change to the network configuration, measurably reducing the impact of hidden terminals on network performance. These case studies show that the hidden terminal detection methods are useful in real-world operational networks and can provide information to help diagnose the root cause of network performance problems.

## 7.2 Conclusions

This thesis set out to answer the question, “can hidden terminals be detected and can their impact on a network be measured using commodity hardware and software?” This thesis has developed, validated, implemented and tested two new methods for detecting hidden terminals as well as a new in-kernel wireless measurement framework in which to implement the methods. The results of laboratory validation as well as real-world operational deployment succeed in answering the original thesis question positively.

As part of implementing the two methods in order to be useful to operators of existing wireless networks, a new approach to wireless measurement was required and so a new in-kernel wireless measurement framework was developed. The new framework allows measurement to be deployed in a cost effective manner over the entirety of the network without the performance penalty or additional costs associated with existing measurement approaches. Reducing the cost and performance impact of measurement allows network operators to deploy measurement more widely than previously possible, increasing the benefits of such measurement.

The combination of the two methods provide valuable insight for operators of wide area 802.11 networks. The network connectivity method can highlight areas in the network that have the possibility for hidden terminal collisions to occur, and it can discount hidden terminal collisions as the cause of poor performance when the network is well connected. The timing method can then measure the impact hidden terminals are having on the network and identify those terminals that are hidden from one another, allowing network operators to make changes to the network, as illustrated in Chapter 6. The use of the two methods in parallel can provide valuable information to the operators of wireless networks.

This thesis concludes that hidden terminals can be detected and their overall effect can be estimated in networks using commodity hardware and software by performing measurement efficiently in-kernel and by using connectivity measurements and packet timing analysis. The methods become useful in an operational context when they are able to be deployed across the entire network without negative performance impact and without additional significant costs associated with previously existing methods for performing wireless measurement. It has also shown through case studies that the ability to detect and measure hidden terminals is useful in a real-world, operational context. The operators of the case study network continue to use the WMP framework and the hidden terminal detection methods as part of their standard network monitoring regime.

## 7.3 Future Work

The hidden terminal detection methods presented in this thesis as well as the WMP kernel measurement framework in general open many opportunities for future research. This section outlines several new areas of research that could now be pursued.

There are many applications for the hidden terminal detection methods outside of what has been presented in this thesis and scope for future work to extend it. The detection of DCF violations could be integrated into wireless receiver hardware and allow the NIC a way to detect the presence of hidden terminal collisions itself and report these to the host operating system. The method could be used by the 802.11 stack to adjust its behaviour in the presence of hidden terminal collisions, either by co-operatively scheduling transmissions or by switching to a TDMA-like protocol, for example.

The hidden terminal detection and measurement methods become even more valuable when applied to increasingly popular ad-hoc and mesh wireless networks. The prevalence and impact of hidden terminals in ad-hoc and mesh networks due to their unplanned nature warrants detailed study. Several studies of the hidden terminal problem in ad-hoc networks exist, for example [34, 22], and the WMP framework could be used to further these by providing further data through the connectivity and timing methods. Ad-hoc and mesh routing protocols could potentially be improved if they were to be made aware of collisions occurring due to the presence of hidden terminals.

The WMP framework for in-kernel wireless measurement and the hidden terminal detection methods presented in this thesis makes longer term studies of hidden terminal behaviour in wireless networks possible. Existing work into simulation of the effect of hidden terminals on throughput and rate control, such as [24], could be extended with data collected from real-world, operational wireless networks.

Bitrate selection algorithms are still an active area of research and a study of the effect of bitrate selection algorithms on hidden terminal collision rates would be a useful piece of future work. For example, if rate selection algorithms are consistently choosing high bitrate encodings resulting in low received signal to noise ratio this may cause terminals to become hidden from one another. This result has been suggested in a recent study of Meraki mesh networks [32] in which the authors state that a lack of data from a wide range of networks reduces the overall understanding of such networks. The WMP framework allows operators of wireless networks based on commodity hardware and software to collect detailed data sets which could be used to improve the availability of such data and hence the understanding of these networks.

More generally, the WMP kernel measurement framework allows for long term, low overhead and wide scale measurement of wireless networks. The WMP framework can be used to collect large amounts of data about the operation of wireless networks in a way that does not significantly impact on the performance of the network. This allows for long term measurement studies and collection of large data sets from real-world networks. Phillips *et al* [45] have used real-world data collected by the WMP kernel measurement framework to publish a study that challenged traditional path loss prediction models. Additional work to collect real-world data sets for use in future research is ongoing.

# Appendix A

# Related Publications

*In-kernel passive measurement of the performance impact of hidden terminals in 802.11 wireless networks* was published in the proceedings of the 8th ACM Symposium on Performance Evaluation of Wireless Ad-Hoc, Sensor, and Ubiquitous Networks, pages 81–88, 2011. It describes the WMP kernel measurement framework and two new methods for hidden terminal detection and measurement [54].

*The Efficacy of Path Loss Models for Fixed Rural Wireless Links* was published in the proceedings of the 12th international conference on Passive and Active Measurement, pages 42–51, 2011. It describes the use of data collected by the WMP kernel measurement framework which was used to challenge traditional path loss prediction models [45].

**Other Publications**

*Using the IEEE 802.11 Frame Check Sequence as a Pseudo Random Number for Packet Sampling in Wireless Networks* was published in the proceedings of the 7th international conference on modelling and optimisation in mobile, ad-hoc and wireless networks, WiOPT'09, pages 552–557 [53].

# Bibliography

[1] HIgh PErformance Radio Local Area Network Type 1: functional specification. Technical report, European Telecommunication Standards Institute, 650 Route des Lucioles, Sophia Antipolis, Valbonne, France,, May 1996.

[2] Kismet. http://www.kismetwireless.net, December 2007.

[3] N. Abramson. The aloha system: another alternative for computer communications. In *AFIPS '70 (Fall): Proceedings of the November 17-19, 1970, fall joint computer conference*, pages 281–285, New York, NY, USA, 1970. ACM.

[4] H. Al-Mefleh and J. M. Chang. Turning hidden nodes into helper nodes in ieee 802.11 wireless lan networks. In *NETWORKING 2008 Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, volume 4982 of *Lecture Notes in Computer Science*, pages 824–835. Springer Berlin / Heidelberg, 2008.

[5] E. Anderson, G. Yee, C. Phillips, D. Sicker, and D. Grunwald. Commodity ar52xx-based wireless adapters as a research platform. Technical Report CU-CS-XXXX-08, University of Colorado at Boulder, Department of Computer Science, Campus Box 430, April 2008.

[6] ANSI/IEEE. *ANSI/IEEE Std 802.11-1997: Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. Institute of Electrical and Electronics Engineers, 1997.

[7] D. Armstrong. *Easing the transition from inpiration to implementaiton: A rapid protoyping platform for wireless medium access control protocols.*

PhD thesis, Department of Computer Science, University of Waikato, 2006.

[8] P. Bahl and V. N. Padmanabhan. Enhancements to the radar user location and tracking system. Technical Report MSR-TR-2000-12, Microsoft Research, One Microsoft Way, February 2000.

[9] P. Bahl and V. N. Padmanabhan. RADAR: an in-building RF-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 775–784. IEEE, 2000.

[10] G. Bernardi, P. Buneman, and M. K. Marina. Tegola tiered mesh network testbed in rural scotland. In *Proceedings of the 2008 ACM workshop on Wireless networks and systems for developing regions*, WiNS-DR '08, pages 9–16, New York, NY, USA, 2008. ACM.

[11] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. Macaw: a media access protocol for wireless lan's. In *SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications*, pages 212–225, New York, NY, USA, 1994. ACM.

[12] F. Borgonovo and L. Fratta. A new technique for satellite broadcast channel communication. In *SIGCOMM '77: Proceedings of the fifth symposium on Data communications*, pages 2.1–2.4, New York, NY, USA, 1977. ACM.

[13] J. Corbet, A. Rubini, and G. Kroah-Hartman. *Linux Device Drivers*, chapter 4, pages 86–90. O'Reilly Media, Inc, 3rd edition, 2005.

[14] M. Durvy, O. Dousse, and P. Thiran. Modeling the 802.11 protocol under different capture and sensing capabilities. In *INFOCOM 2007. 26th Annual IEEE Conference on Computer Communications*, pages 2356–2360. IEEE, 2007.

[15] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. In *IEEE/ACM Transactions on Networking*, volume 1, pages 397–413, August 1993.

[16] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla. The impact of multihop wireless channel on tcp throughput and loss. In *INFOCOM*

*2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1744–1753, March 2003.

[17] C. L. Fullmer and J. J. Garcia-Luna-Aceves. Floor acquisition multiple access (fama) for packet-radio networks. In *SIGCOMM '95: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pages 262–273, New York, NY, USA, 1995. ACM.

[18] C. L. Fullmer and J. J. Garcia-Luna-Aceves. Solutions to hidden terminal problems in wireless networks. *SIGCOMM Comput. Commun. Rev.*, 27(4):39–49, 1997.

[19] L. Gavrilovska. Cross-layering approaches in wireless ad hoc networks. *Wireless Personal Communications*, 37(3-4):271–290, May 2006.

[20] S. Gollakota and D. Katabi. Zigzag decoding: combating hidden terminals in wireless networks. In *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pages 159–170, New York, NY, USA, 2008. ACM.

[21] E. Haghani, M. N. Krishnan, and A. Zakhor. Adaptive carrier-sensing for throughput improvement in IEEE 802.11 networks. In *Global Telecommunications Conference (GLOBECOM 2010)*, pages 1–6. IEEE, 2010.

[22] T. Han and W. Lu. An improvement of maca in alleviating hidden terminal problem in adhoc networks. In *Proceedings of the 5th International Conference on Wireless communications, networking and mobile computing*, WiCOM'09, pages 2870–2873, Piscataway, NJ, USA, 2009. IEEE Press.

[23] K. K. He. Kernel korner: why and how to use netlink socket. *Linux Journal*, 2005(130):11, February 2005.

[24] K.-L. Hung and B. Bensaou. Throughput analysis and rate control for ieee 802.11 wireless lan with hidden terminals. In *MSWiM '08: Proceedings of the 11th international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 140–147, New York, NY, USA, 2008. ACM.

[25] L. Iannone, K. Kabassanov, and S. Fdida. The real gain of cross-layer routing in wireless mesh networks. In *REALMAN '06: Proceedings of the second international workshop on Multi-hop ad hoc networks: from theory to reality*, pages 15–22, New York, NY, USA, 2006. ACM Press.

[26] IEEE. *IEEE Std 802.11k-2008: Amendment 1: Radio Resource Measurement of Wireless LANs*. Institute of Electrical and Electronics Engineers, 2008.

[27] A. P. Jardosh, K. N. Ramachandran, K. C. Almeroth, and E. M. Belding-Royer. Understanding link-layer behavior in highly congested ieee 802.11b wireless networks. In *E-WIND '05: Proceeding of the 2005 ACM SIGCOMM workshop on Experimental approaches to wireless network design and analysis*, pages 11–16, New York, NY, USA, 2005. ACM Press.

[28] S. Khurana, A. Kahol, and A. P. Jayasumana. Effect of hidden terminals on the performance of ieee 802.11 mac protocol. *Local Computer Networks, Annual IEEE Conference on*, 0:12, 1998.

[29] L. Kleinrock and F. Tobagi. Packet Switching in Radio Channels: Part I–Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics. *Communications, IEEE Transactions on [legacy, pre-1988]*, 23(12):1400–1416, 1975.

[30] A. Kochut, A. Vasan, A. Shankar, and A. Agrawala. Sniffing out the correct physical layer capture model in 802.11b. In *Network Protocols, 2004. ICNP 2004, Proceedings of the 12th IEEE International Conference on*, pages 252 – 261, October 2004.

[31] K. Kosek, M. Natkaniec, and L. Vollero. Thorough analysis of 802.11e star topology scenarios in the presence of hidden nodes. In *NETWORKING 2008 Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, volume 4982 of *Lecture Notes in Computer Science*, pages 792–803. Springer Berlin / Heidelberg, 2008.

[32] K. LaCurts and H. Balakrishnan. Measurement and analysis of real-world 802.11 mesh networks. In *Proceedings of the 10th annual conference on*

*Internet measurement*, IMC '10, pages 123–136, New York, NY, USA, 2010. ACM.

[33] J. Lee, W. Kim, S.-J. Lee, D. Jo, J. Ryu, T. Kwon, and Y. Choi. An experimental study on the capture effect in 802.11a networks. In *WinTECH '07: Proceedings of the the second ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*, pages 19–26, New York, NY, USA, 2007. ACM.

[34] F. Y. Li, A. Kristensen, and P. Engelstad. Passive and active hidden terminal detection in 802.11-based ad-hoc networks. In *Proc. IEEE Infocom Conference*, 2006.

[35] X. Lu, G. Fan, and R. Hao. A dynamic token passing mac protocol for mobile ad hoc networks. In *IWCMC '06: Proceedings of the 2006 international conference on Wireless communications and mobile computing*, pages 743–748, New York, NY, USA, 2006. ACM.

[36] H. Ma, R. Vijayakumar, S. Roy, and J. Zhu. Optimizing 802.11 wireless mesh networks based on physical carrier sensing. *IEEE/ACM Trans. Netw.*, 17(5):1550–1563, Oct. 2009.

[37] P. Malindi and T. Kahn. Enabling broadband rural networking. *Broadband Communications, Information Technology and Biomedical Applications, International Conference on*, 0:82–88, 2008.

[38] D. Malone, D. J. Leith, and I. Dangerfield. Inferring queue state by measuring delay in a wifi network. In *Traffic Monitoring and Analysis*, volume 5337 of *Lecture Notes in Computer Science*, pages 8–16. Springer Berlin / Heidelberg, May 2009.

[39] S. M. Mishra, J. Hwang, R. Moazzami, L. Subramanian, and T. Du. Economic analysis of networking technologies for rural developing regions. In *1st Workshop on Internet and Network Economics*, 2005.

[40] W. M. Moh, D. Yao, and K. Makki. Analyzing the hidden-terminal effects and multimedia support for wireless lan. *Computer Communications*, 23(10):998 – 1013, 2000.

[41] Network Simulator 2 - ns2. http://www.isi.edu/nsnam/ns/.

[42] A. M. Nungu, B. Pehrson, and N. Genesis. Serengeti broadband. In *Proceedings of the second ACM SIGCOMM workshop on Networked systems for developing regions*, NSDR '08, pages 37–42, New York, NY, USA, 2008. ACM.

[43] S. Papanastasiou, L. M. MacKenzie, and M. Ould-Khaoua. Reducing the degrading effect of hidden terminal interference in manets. In *MSWiM '04: Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 311–314, New York, NY, USA, 2004. ACM.

[44] S. Papanastasiou, M. Ould-Khaoua, and V. Charissis. The effect of the rts/cts handshake on tcp. *Advanced Information Networking and Applications Workshops, International Conference on*, 2:940–946, 2007.

[45] C. Phillips, S. Raynel, J. Curtis, S. Bartels, D. Sicker, D. Grunwald, and T. McGregor. The efficacy of path loss models for fixed rural wireless links. In *Proceedings of the 12th international conference on Passive and active measurement*, PAM'11, pages 42–51, Berlin, Heidelberg, 2011. Springer-Verlag.

[46] Radiotap. http://www.radiotap.org.

[47] G. Rahmatollahia, S. Gallerb, J. Schroederc, K. Jobmannd, and K. Kyamakyae. Propagation Delay Based Positioning Using IEEE 802.11b Signals. *Proceedings of the 3rd Workshop on Positioning, Navigation and Communication (WPNC '06)*, pages 169–177, 2006.

[48] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168 (Proposed Standard), Sept. 2001. Updated by RFCs 4301, 6040.

[49] S. Ray, J. Carruthers, and D. Starobinski. Rts/cts-induced congestion in ad hoc wireless lans. In *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, volume 3, pages 1516–1521, March 2003.

[50] S. Ray, J. Carruthers, and D. Starobinski. Evaluation of the masked node problem in ad hoc wireless lans. In *Mobile Computing, IEEE Transactions on*, volume 4, pages 430–442, Sept.-Oct. 2005.

[51] S. Ray, D. Starobinski, and J. B. Carruthers. Performance of wireless networks with hidden nodes: a queuing-theoretic analysis. *Computer Communications*, 28(10):1179 – 1192, 2005. Performance issues of Wireless LANs, PANs and ad hoc networks.

[52] M. Raya, J.-P. Hubaux, and I. Aad. Domino: a system to detect greedy behavior in ieee 802.11 hotspots. In *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 84–97, New York, NY, USA, 2004. ACM.

[53] S. Raynel, A. McGregor, and M. Jorgensen. Using the ieee 802.11 frame check sequence as a pseudo random number for packet sampling in wireless networks. In *WiOPT'09: Proceedings of the 7th international conference on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, pages 552–557, Piscataway, NJ, USA, 2009. IEEE Press.

[54] S. M. Raynel and T. J. McGregor. In-kernel passive measurement of the performance impact of hidden terminals in 802.11 wireless networks. In *Proceedings of the 8th ACM Symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, PE-WASUN '11, pages 81–88, New York, NY, USA, 2011. ACM.

[55] L. G. Roberts. Aloha packet system with and without slots and capture. *SIGCOMM Comput. Commun. Rev.*, 5(2):28–42, 1975.

[56] N. Santhapuri, R. R. Choudhury, J. Manweiler, S. Nelakuduti, S. Sen, and K. Munagala. Message in message mim: A case for reordering transmissions in wireless networks. In *In HotNets*, 2008.

[57] A. Schulman, D. Levin, and N. Spring. On the fidelity of 802.11 packet traces. In *Passive and Active Network Measurement*, volume 4979/2008 of *Lecture Notes in Computer Science*, pages 132–141. Springer Berlin / Heidelberg, April 2008.

[58] O. Sharon and E. Altman. An efficient polling mac for wireless lans. *IEEE/ACM Trans. Netw.*, 9(4):439–451, August 2001.

[59] A. Sheth, C. Doerr, D. Grunwald, R. Han, and D. Sicker. Mojo: a distributed physical layer anomaly detection system for 802.11 wlans. In

*MobiSys '06: Proceedings of the 4th international conference on Mobile systems, applications and services*, pages 191–204, New York, NY, USA, 2006. ACM.

[60] K.-P. Shih, W.-H. Liao, H.-C. Chen, and C.-M. Chou. On avoiding rts collisions for ieee 802.11-based wireless ad hoc networks. *Computer Communications*, 32(1):69 – 77, 2009.

[61] F.-J. Simo-Reigadas, A. Martinez-Fernandez, F.-J. Ramos-Lopez, and J. Seoane-Pascual. Modeling and optimizing ieee 802.11 dcf for long-distance links. *IEEE Transactions on Mobile Computing*, 9:881–896, 2010.

[62] B. Singh, N. Mani, S. Kadyan, R. K. Shukla, and A. Singh. A high performance point-to-point rural wi-fi and wi-max access networks in developing regions. *Network Applications, Protocols and Services, International Conference on*, 0:203–208, 2010.

[63] L. Subramanian. Rethinking wireless in the developing world. In *HotNets 2006: Proceedings of the 5th Workshop on Hot Topics in Networks*, 2006.

[64] C. Thorpe, S. Murphy, and L. Murphy. IEEE802.11k enabled adaptive physical carrier sense mechanism for wireless networks (K-APCS). In *Proceedings of the 4th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, PM2HW2N '09, pages 209–215, New York, NY, USA, 2009. ACM.

[65] F. Tobagi and L. Kleinrock. Packet Switching in Radio Channels: Part II–The Hidden Terminal Problem in Carrier Sense Multiple-Access and the Busy-Tone Solution. *Communications, IEEE Transactions on*, 23(12):1417–1433, 1975.

[66] F. Tobagi and L. Kleinrock. Packet Switching in Radio Channels: Part III–Polling and (Dynamic) Split-Channel Reservation Multiple Access. *Communications, IEEE Transactions on [legacy, pre-1988]*, 24(8):832–845, 1976.

[67] O. W. Visser. Localisation in large-scale outdoor wireless sensor networks. Master's thesis, Delft University of Technology, August 2005.

[68] C. Ware, J. Judge, J. Chicharo, and E. Dutkiewicz. Unfairness and capture behaviour in 802.11 adhoc networks. In *IEEE International Conference on Communications, 2000*, volume 1, pages 159–163 vol.1, 2000.

[69] C. Ware, T. Wysocki, and J. Chicharo. Hidden terminal jamming probems in ieee 802.11 mobile ad hoc networks. *IEEE International Conference on Communications*, 1:261–265, 2001.

[70] C. Wu and V. Li. Receiver-initiated busy-tone multiple access in packet radio networks. In *SIGCOMM '87: Proceedings of the ACM workshop on Frontiers in computer communications technology*, pages 336–342, New York, NY, USA, 1988. ACM.

[71] K. Xu, M. Gerla, and S. Bae. Effectiveness of rts/cts handshake in ieee 802.11 based ad hoc networks. *Ad Hoc Networks*, 1(1):107–123, July 2003.

[72] S. Xu and T. Saadawi. Revealing the problems with 802.11 medium access control protocol in multi-hop wireless ad hoc networks. *Comput. Netw.*, 38(4):531–548, 2002.

[73] J. Yeo, M. Youssef, and A. Agrawala. A framework for wireless lan monitoring and its applications. In *WiSe '04: Proceedings of the 3rd ACM workshop on Wireless security*, pages 70–79, New York, NY, USA, 2004. ACM.

[74] J. Yeo, M. Youssef, T. Henderson, and A. Agrawala. An accurate technique for measuring the wireless side of wireless networks. In *WiTMeMo '05: Papers presented at the 2005 workshop on Wireless traffic measurements and modeling*, pages 13–18, Berkeley, CA, USA, 2005. USENIX Association.

[75] J. Zhu, X. Guo, L. L. Yang, W. S. Conner, S. Roy, and M. M. Hazra. Adapting physical carrier sensing to maximize spatial reuse in 802.11 mesh networks. *Wireless Communications and Mobile Computing Journal*, 4:933–946, 2004.