MODELS AND SYSTEMS FOR UNDERSTANDING WIRELESS INTERFERENCE

by

Shravan Rayanchu

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2012

Date of final oral examination: 06/07/2012

The dissertation is approved by the following members of the Final Oral Committee:

Suman Banerjee, Associate Professor, Computer Sciences
Aditya Akella, Associate Professor, Computer Sciences
Paul Barford, Professor, Computer Sciences
Ranveer Chandra, Senior Researcher, Microsoft Research
Stark Draper, Assistant Professor, Electrical and Computer Engineering

To my parents, Bhavani and Vaikuntam Rayanchu,
and my sister, Manisha Rayanchu.

ACKNOWLEDGMENTS

During my graduate school, I have had the good fortune of meeting and learning from many wonderful people, and I am grateful to all of them. First and foremost, I want to thank my advisor, Suman Banerjee. I am appreciative of his generosity with time, advice, ideas, and am grateful to him for his guidance during my graduate studies. Suman gave me an incredible amount of freedom to work on problems of my choice and to figure things out at my own pace. And besides being a great advisor, he has made himself available for all my problems, both professional and personal, throughout my graduate school. Thank you, Suman.

I consider myself lucky to have been mentored by some exceptional minds. I would like to thank Ranveer Chandra for his guidance and support. His valuable advice has helped shape some crucial aspects of my work. I would also like to thank my mentors Ram Ramjee and Sudipta Sengupta who have always been available for helpful advice and guidance. Likewise, I would also like to thank Aditya Akella, Paul Barford and Stark Draper for their insightful suggestions and comments that greatly improved this thesis. Thanks also to Alec Wolman, Stefan Saroiu and Victor Bahl for a very enjoyable summer I spent at Microsoft Research.

I have enjoyed working with, and have had the support of, some excellent colleagues and friends at the WiNGS lab. I would like to thank Arunesh Mishra for mentoring me in the early stages of my research and for introducing me to the joys of experimental research. I have greatly benefitted by his infectious enthusiasm for work and his constant emphasis for working on practical problems. I am also indebted to my colleague and close friend Vivek Shrivastava for always being available, be it for bouncing off ideas, help with the testbed set up, debugging weird issues, or simply, for being there to let some steam off with a cold beer when nothing seemed to work. Likewise, I would like to thank Sayandeep Sen for the camaraderie and support, especially during the later years of my graduate school. I will fondly remember the innumerable discussions I have had with him over coffee at the Union South. I also want to

express my thanks to other students in the lab — Dheeraj Agrawal, Sharad Saha, Ashutosh Shukla, Ashish Patro, Jongwon Yoon, Joshua Hare, Vladimir Brik, Tan Zhang, Mike Griepentrog, Lance Hartung and Yadi Ma — for fostering a conducive environment for research and for being great colleagues.

Apart from the students in the lab, I have the company of several good friends in the department with whom I have spent some fun times. Adwait Tumbde, Siddharth Barman, Pradeep Tamma, Shreepadma Venugopalan, Jayaram Bobba, Cindy Rubio, Ashok Anand and Theo Benson have provided great company. Outside of the department, Namita Azad, Megha Desai, Prachi Singh, and Aparna Vidyasagar have helped create some wonderful memories from my time in Madison.

I would not be here without some exceptional people to cheer me and support me every step of the way. I want to thank Abhijeet Kumar, Prabhanshu Shekhar, Anand Sinha and Vinay Choudhary for being great companions, and most importantly, for always being available to listen to my rants! Special thanks to my close friend, Abhinav Sarje. Besides being exceptionally patient with me, he has been a source of unwavering support for the past ten years. I will forever cherish his friendship. Going back in time, I am grateful to my teachers in school and at IIT Guwahati who have played a crucial role in my education. I also want to thank my friends for the past fourteen years, Sandeep Deshpande and Vinay Bandaru. Their love and generosity is only paralleled by their exceptional sense of humor.

My heartfelt thanks to Akanksha Baid, without whom this journey wouldn't have been the same. I thank her for six memorable years of friendship, and for always being there to brighten my day. She means the world to me.

My family always makes me feel like I am the luckiest person on earth. Tata, Ammamma, Nanamma, Arupinni, Jayakka, Vasumama, my cousins and the rest of my extended family have showered me with much love and affection, and I am thankful for having spent my childhood amidst such wonderful people.

I owe a special debt of gratitude to my parents, Bhavani and Vaikuntam Rayanchu, and my little sister, Manisha. They have loved me and cared for me unconditionally all my life. And simply put, they are the reason I have been able to get this far. I know words aren't enough, but I sincerely thank them for

all the sacrifices they have made and everything they have done for me. I love them and I dedicate this dissertation to them.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

MODELS AND SYSTEMS FOR UNDERSTANDING WIRELESS INTERFERENCE

Shravan Rayanchu

Under the supervision of Associate Professor Suman Banerjee
At the University of Wisconsin-Madison

With the prolific growth in the usage and the number of wireless devices, the problem of designing high performance and reliable wireless networks is of great importance today. A crucial step forward in designing such networks is to improve our understanding of *wireless interference* — a phenomenon that remains the primary source of performance bottlenecks and unpredictability in today's wireless networks. To this end, this dissertation makes contributions in developing systems and models that help us better understand interference in wireless networks.

We start by studying wireless losses. Wireless losses can be due to weak signal at the receiver, or interference from other wireless devices such as WiFi devices or non-WiFi RF devices. To isolate the losses emanating from interference due to WiFi devices, we design COLLIE, a system that helps distinguish between losses due to weak signal and those due to WiFi interference. We then focus our attention to the problem of non-WiFi interference. Since WiFi cards do not have the ability to decode a non-WiFi device's transmission, we designed Airshark a software system that runs on top of a mainstream wireless card and helps detect different non-WiFi devices. We then designed WiFiNet, a system that helps detect the exact source of interference, quantify the impact of such interference, and also help physically pin point the non-WiFi interferer's location. All these systems are based on post-mortem analysis of wireless losses. In the last part of this thesis, we also explore an alternative method of modeling wireless interference in the context of links using flexible channels. These models are predictive in nature and help us understand the impact of wireless interference before the event of a packet loss.

We believe that the models and systems presented in this thesis are useful in understanding wireless interference in today's networks as our solutions

can be natively implemented in present-day WiFi hardware. We hope that our contributions would serve as useful building blocks in designing more sophisticated tools to better understand wireless interference and pave way for building future wireless networks that are more reliable, predictable and manageable.

Suman Banerjee

1 INTRODUCTION

Wireless networks have experienced a phenomenal growth over the last decade [111, 154], and WiFi has become the predominant technology for communication in indoor wireless environments. More recently, we have also witnessed a tremendous growth in the number and usage of mobile devices [161], largely attributable to concomitant maturation of mobile ecosystems such as iOS [15] and Android [14] systems. These mobile systems are abundant with rich, media-centric applications that require low latency and high bandwidth [48] [153]. Naturally, the growing usage of such demanding mobile applications has led the users to expect *wire-like* performance from WiFi networks *i.e.*, users expect WiFi networks to be *fast*, and they expect the networks to be *reliable*. However, unlike their wired brethren, wireless networks do not yet offer a predictable performance, and offer significantly lower throughputs than their wired counterparts. Although, newer standards like 802.11n are capable of offering raw data rates of around 300 Mbps, the actual throughputs in practice remain considerably lower [147].

A primary reason for WiFi performance degradation is the *lossy* nature of WiFi links (WiFi transmitter and receiver pairs). For the purposes of this thesis, we classify packet losses on WiFi links into two broad categories:

- *Weak signal*: A “weak signal" scenario leads to packet losses when the received signal strength of the transmitted packet is not strong enough for it to be decoded successfully at the receiver. Channel impairments such as signal attenuation, shadowing and short-term channel fading due to multipath propagation [129] are typically responsible for such weak signal scenarios.

- *Interference*: A WiFi transmission can suffer from interference at the receiver when there is a simultaneous transmission from another transmitter (interferer). Depending on the magnitude of signal strength of the interferer at the receiver, such interference can result in packet losses.

For the "weak signal" scenario, whether the received signal strength (or more accurately, the signal strength to noise ratio, SNR) of the transmitter is enough for the packet to be decoded successfully primarily depends on the receive sensitivity [133] of the wireless card, which is a function of the PHY data rate at which the packet was modulated at. Such packet losses attributable to weak signal can happen frequently, either because the client is too far away from the AP, or because of aggressive data-rate adaptation algorithms [28] that attempt to operate a wireless link at the highest rate possible in order to maximize throughput and overall system capacity.

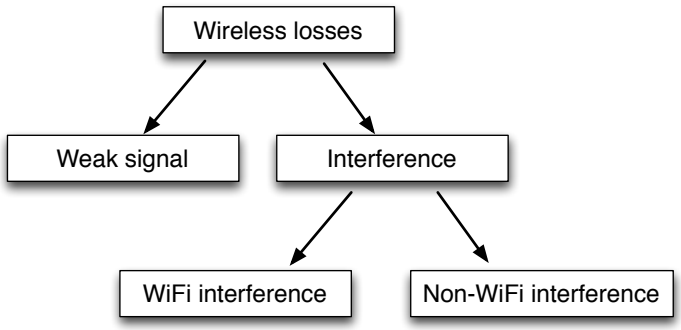

Figure 1.1: For the purposes of this thesis, we broadly classify wireless packet losses into two categories: (a) losses due to weak signal and (b) losses due to wireless interference. Wireless interference experienced by a link can be because of (a) WiFi to WiFi interference or (b) non-WiFi to WiFi interference.

Wireless interference experienced by WiFi links on the other hand can be caused due to a number of reasons. A wireless network using 802.11 a/b/g/n operates in an unlicensed spectrum, which can be populated by a plethora of WiFi transmitters such as other WiFi Access Points (APs) and clients, as well as many other non-WiFi wireless devices such as Bluetooth devices, ZigBee devices, cordless phones and microwave ovens. Simultaneous transmissions from these devices can interfere at the WiFi receiver and result in packet losses, thereby decreasing the observed link throughputs. In this thesis, we broadly categorize such interference experienced by a WiFi link into: (i) *WiFi to WiFi interference i.e.*, interference experienced by a WiFi link due to other WiFi transmitters and

(ii) *non-WiFi to WiFi interference i.e.*, interference experienced by a WiFi link due to other non-WiFi wireless transmitters that use the same spectrum (e.g., a 2.4 GHz channel) as WiFi links. Figure 1.1 summarizes this broad category of reasons for a wireless packet loss.

Impact of interference

A classic example of WiFi to WiFi interference is a scenario where simultaneous transmissions from two WiFi devices can result in packet collisions at the WiFi receiver(s) and degrade system throughput, a scenario commonly referred to as the *hidden terminal* case [27]. Another example of throughout degradation in presence of WiFi devices is the case of *exposed terminals*, where two WiFi transmitters may carrier sense each other and defer the their transmissions unnecessarily [27, 148]. Similarly, interference from non-WiFi devices such as analog cordless phones or microwave ovens can cause packet losses at the WiFi receiver or can cause the WiFi transmitters to endlessly defer their transmissions [139] leading to a complete loss of connectivity.

Non-WiFi interference can be particularly damaging as most non-WiFi devices are "agnostic" of other WiFi devices operating in the vicinity. For example, in case of two WiFi transmitters that can potentially interfere with each other, WiFi's constituent standard, 802.11, has carrier sensing and back-off mechanisms [67] in place to avoid such WiFi to WiFi interference. However, many non-WiFi devices do not follow such interference avoidance mechanisms (e.g., analog cordless phones transmit energy continuously), and in some cases (e.g., devices like microwave ovens) the transmitted energy is simply "signal leakage" that cannot be avoided [81, 150]. Consequently, most non-WiFi devices being agnostic of WiFi transmissions, do not back-off to WiFi transmissions in progress and severely interfere at the WiFi receiver. Furthermore, even 802.11's interference avoidance mechanisms only take other WiFi transmitters into account. That is, WiFi devices cannot "decode" transmissions from other non-WiFi devices and therefore cannot identify and react to non-WiFi interference.

It is a well known fact that packet losses in 802.11 can happen due to the above stated reasons — weak signal, or WiFi to WiFi interference, or non-WiFi to WiFi interference. However, discerning the exact cause of a packet loss, once it occurs, is known to be quite difficult. Attributing the correct cause for a packet loss is important as this decision triggers the choice for operational parameters to be used on each wireless link and provides a crucial input to interference mitigation strategies employed, thus affecting the overall performance of the wireless network. For example, if it determined that a weak signal scenario is responsible for packet losses, then simply deploying more WiFi APs to ensure a better coverage [112], or increasing the transmit power at the WiFi transmitter [126], or in yet another case reducing the PHY data rate of the transmitter might alleviate the problem [163]. Similarly, in the case of interference, a variety of interference mitigation strategies such as channel assignment [107], RTS-CTS mechanisms coupled with rate adaptation [163], packet scheduling [148], conflict-aware transmit power control mechanisms [126] can be used.

Determining the correct strategy to mitigate wireless wireless losses, however, is only possible after knowing the underlying cause. Hence, there is a need for systems that determine the exact cause of a wireless loss and attribute it to weak signal, or interference from WiFi devices, or interference from non-WiFi devices. Such systems would help identify the underlying cause for performance degradation of various links in the network and help improve their performance. Next, we motivate the need for such systems with the help of an example shown below.

A Motivating scenario

Consider an example of an enterprise wireless LAN (WLAN) shown in the Figure 1.2. This WLAN consists of 5 WiFi APs, all of which are connected to a central WLAN controller over an Ethernet backplane. Such an architecture is typical of many enterprise WLANs that are deployed in practice [148, 149]. In this WLAN, are a total of 6 clients — WiFi-enabled laptops, smartphones and

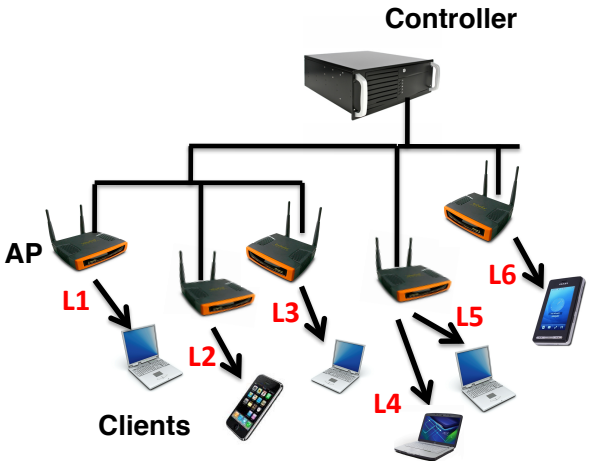

Figure 1.2: An example enterprise WLAN with a central controller, 5 WiFi Access Points (APs) and 6 clients — WiFi-enabled laptops, smartphones and tablets. Each client is associated to one AP, therefore there are 6 WiFi links in the network. These links are labelled L1, L2, . . ., L6.

tablets. Each client is associated to one AP, *i.e.*, there are 6 AP-client links in the network. These links are labelled L1, L2, . . ., L6.

Figure 1.3 shows the performance of each of the individual WiFi links in this network[1]. In this example, only links L5 and L6 have a good delivery ratio (close to 1). Rest of the links do not have very good delivery ratios. We note that measuring link delivery ratios is not difficult. For example, one can deploy wireless sniffers that monitor the packets transmitted by each link, infer the whether each packet was successfully transmitted or not (based on the receipt of acknowledgment packets or retries [123]) and measure the link delivery ratio. Several existing solutions (commercial systems as well as research prototypes) are able to use such mechanisms and measure the link delivery ratio. However, these existing systems are not able to explain and comprehensively point out the underlying reason behind a WiFi link's performance (*i.e.*, the delivery ratio).

[1]In this example, we use the link delivery ratio, a popular approach to measure a WiFi link's MAC layer performance. We note that wireless interference resulting in packet losses can also affect a number of higher layer quality-of-service metrics such as delay or jitter.

Specifically, if the delivery ratio of a link is low, it is important to identify *why* the performance of a particular WiFi link is low.

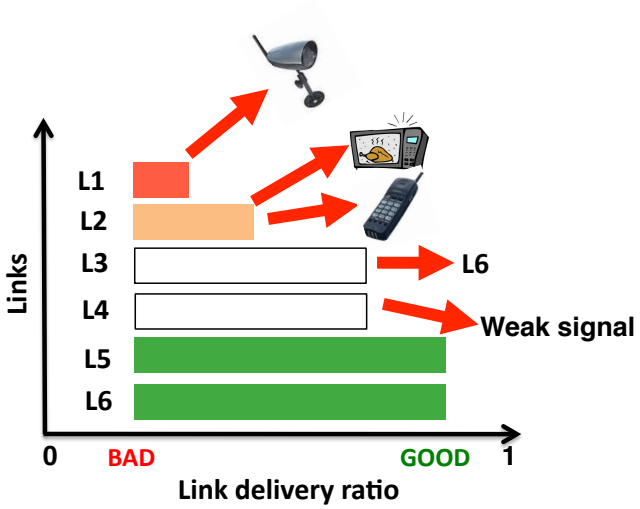

Figure 1.3: The overall performance of the WLAN shown in Figure 1.2. We use link delivery ratio to measure the performance of individual links. The figure also shows the reason behind under performance of each link (e.g., interference from a non-WiFi device, or interference from a WiFi device, or a weak signal problem). Our objective in this thesis is to develop systems and models that help us identify such interferers and quantify their impact.

In this example WLAN, it turns out that links L1 . . . L4 do not have good delivery ratios because of a variety of reasons as shown in Figure 1.3 — Link L1 has a poor delivery ratio as it is suffering from interference due to a non-WiFi interferer, a wireless video camera. Link L2 on the other hand is suffering from interference due to two non-WiFi devices, a microwave oven and a cordless phone. Link L3 is suffering from interference due to a link L6 (WiFi to WiFi interference). Link L4 is not suffering from any interference, however, the link is suffering from a "weak signal" problem *i.e.*, L4's receiver (a laptop) is far away from the transmitter (AP) and is resulting in some of the packets getting lost.

We note that the loss mitigation strategy in each of the above cases is *dependent* on the underlying reason for the loss. For example, to improve link

L1's delivery ratio, the only solution might be to change the L1 to a different channel to avoid interference from the video camera that transmits energy continuously on fixed channel. Whereas in the case of L2, an appropriate packet scheduling algorithm [148] can be invoked to mitigate losses due to a microwave oven that exhibits an ON-OFF transmission pattern [81, 150]. Since L3 is being interfered by L6, a variety of WiFi interference mitigation mechanisms e.g., link scheduling [148] or channel assignment [107] can be used to make the links non-conflicting. In the case of L4, which is suffering from a weak signal problem, increasing the transmit power on L4's AP might improve the link delivery ratio.

It is important to note that the correct interference mitigation strategy in each case can be employed only after knowing the underlying cause for the wireless loss. Hence there is a need for systems that help identify the underlying reason for a wireless packet loss and estimate the extent of wireless interference experienced by the network, thereby helping us better understand and manage interference in today's wireless LANs. To this end, in this thesis, we focus on the problem of wireless interference detection and quantification.

Interference estimation under the constraints of today's WiFi hardware: challenges and opportunities

While it is desirable that we are able to estimate interference (from both WiFi and non-WiFi devices) in present-day WLANs, a solution that enables such estimation on top of today's WiFi hardware has to tackle several challenges. WiFi interference estimation has been an active research topic and several solutions, some of which work under the constraints of WiFi hardware, have been proposed (Chapter 6). Below, we explain the challenges in interference estimation mostly in context of non-WiFi interferers. Subsequently, we outline the basic idea behind the solutions developed in this dissertation for interference estimation.

Why is it hard to estimate non-WiFi interference with current WiFi hardware?

Estimating interference from non-WiFi devices using current WiFi cards is challenging because of several reasons:

- *Challenge #1.* In order to estimate the impact of a non-WiFi interferer, an AP or a client employing a WiFi card has to be able to *detect* the presence of this non-WiFi interferer. Fundamentally, however, WiFi cards are not designed to decode non-WiFi device transmissions. That is, current circuitry used in WiFi cards hosts digital MAC and baseband engines that can detect and decode *only* WiFi transmissions.
  Implication #1. The above restriction implies that non-WiFi transmissions cannot be decoded by WiFi cards as is. Further, the absence of such decoder modules implies that several prior research efforts that advocate usage of protocol specific demodulators or decoders [98] are not applicable when employing WiFi hardware.

- *Challenge #2.* Current WiFi cards do not expose raw signal information. For example, raw signal information in the form of baseband I/Q samples from the output of the analog-to-digital converter (ADC) in the RF front end are not exposed by today's WiFi hardware. Traditionally, only RSSI (received signal strength) information about the packet, or more recently, RSSI per sub-carrier information has been exposed by WiFi cards.
  Implication #2. Lack of such signal information implies that several approaches in the literature that are able to extract information about the original interferer by entirely reconstructing the interferer's signal [98], or by using cyclostationary detection approaches [66] that work with the raw I/Q samples are not applicable. Thus, *one has to develop non-WiFi device detection approaches that work solely on these power samples*. That is, these mechanisms should work despite the absence of phase and modulation properties of the signal.

- *Challenge #3.* Current WiFi cards are designed to operate on a narrow band of spectrum (e.g., 20 MHz or 40 MHz). This is because, in a typical

WiFi receiver based on an Atheros 802.11 chip [141], the center frequency and channel bandwidth are determined by the frequency synthesizer and the phase locked loop (PLL) in the RF front end circuitry. Both frequency synthesizer and PLL are fed by a reference clock that is driven by a 20 (or a 40 MHz) crystal oscillator [141]. Thus, the amount of spectrum that can be sampled by today's WiFi cards is limited to a maximum of 20 (or 40) MHz.

Implication #3. Several prior approaches that make use of wide-band radios [125] or sophisticated signal analyzers [1, 66] that can sample a wide-band of 80−100 MHz (e.g., the entire 2.4 GHz band) are not applicable. An important implication is that the solutions that detect non-WiFi device transmissions must be able to cope with lost spectrum information — power samples from parts of the spectrum not monitored by a WiFi card. In particular, developed solutions must be able to detect non-WiFi device transmissions that can potentially occupy a much wider band. For example, frequency hoppers such as bluetooth or cordless phone devices spread their transmissions across the entire 80 MHz band. Non-WiFi detection solutions must be able to detect such frequency hoppers despite not being able to capture all their transmissions.

- *Challenge #4.* By design, WiFi cards offer reduced sampling resolutions in both time and frequency domains. For example, WiFi cards have a resolution bandwidth [1] of 312.5 kHz, equal to sub-carrier spacing in 802.11. Processing the signal in each of the 802.11 sub-carriers is a part of the regular OFDM decoding circuitry implemented in today's WiFi cards. Information about power in each of these sub-carriers is exposed by current wireless cards leading to resolution bandwidth of 312.5 kHz. Current WiFi cards also have a low sampling rate in the time domain. For example, current Atheros WiFi cards offer ~2.5k samples/sec. While, regular WiFi cards are capable of decoding each OFDM symbol (~4µs in duration), many WiFi cards (e.g., Atheros wireless cards) employ circuitry to expose only time-averaged information about the signal such RSSI or RSSI-per sub-carrier at a coarse-grained time granularity (e.g., inter-

sample duration of ~120µs). Such an time averaging is typically employed for carrier sensing and radar detection capabilities [68].
Implication #4. Several prior research efforts [66, 98] that assume high sampling resolutions (e.g., as low as 1 kHz frequency domain resolution) offered by channel sounders, commercial signal analyzers [3] (160k samples/sec) or software radios (8 million samples/sec) are not applicable. Non-WiFi device detection capabilities developed on top of commodity WiFi hardware must employ techniques that work with such low sampling resolution.

- *Challenge #5.* Despite limited power information exposed by WiFi cards, they should be able to *uniquely identify* a non-WiFi device, especially in the presence of multiple devices of the same type.
  Implication #5. Since the detection procedures must solely operate on power samples, non-WiFi device detection can be more challenging in the presence of *multiple, simultaneously operating devices* as their transmission characteristics can potentially overlap. Further, in some cases where similar devices are under operation, it becomes difficult to distinguish between these devices. For example, transmission power characteristics of two cordless phones can be potentially identical (even the amount of received power can be similar if the devices are at an equal distance from the WiFi card and if they use the same transmit power). Thus, device detection procedures must be able to uniquely identify different non-WiFi devices (even multiple devices of the *same type*) despite similar power characteristics.

- *Challenge #6.* A system based on WiFi cards must be able to *estimate* a non-WiFi device's interference impact (in the presence of multiple interferers) and also be able to *physically locate* the device despite not knowing the transmit power of the device.
  Implication #6. Using only information built from power samples, the solution must first uniquely identify different non-WiFi devices, segregate their transmissions, and then identify the interference impact of each such device. Further, in typical localization procedures (e.g., in WiFi

localization procedures such as [25, 29, 140, 166]), multiple detectors have to to *detect the same transmission* at different signal strengths. This is easy in WiFi localization because different WiFi detectors decode the same wireless frame and use the frame's identity to ensure sameness. A core challenge that we have to solve is to use different WiFi detectors and determine which transmissions correspond to a single transmission instance from the same non-WiFi device. Further, most of the prior localization approaches assume that target device's transmit power is known [25, 29, 140, 166]. Our solution has to able to localize each non-WiFi device without knowing its actual transmit power.

Is there an opportunity for interference estimation with current WiFi hardware?

While current WiFi cards impose certain limitations as explained above, in this dissertation, we will show that is possible to estimate interference from both WiFi and non-WiFi devices using such commonly available WiFi hardware. Here, we explain the basic idea behind the solutions developed in this dissertation for interference estimation, particularly in the context of interference from non-WiFi devices. Our solutions work well despite the constraints imposed by commodity WiFi hardware and across a wide range of non-WiFi devices, as they only rely on *received power of transmissions* — a property that is generic, and is inherent to all wireless communication technologies as we explain next.

As explained previously, WiFi cards only expose limited about of power information about the signal. Typically, this information is averaged across the entire channel and for every packet (e.g., RSSI per packet), or it is provided at a slightly finer granularity (e.g., RSSI per sub-carrier time averaged across few OFDM symbols) such as in emerging WiFi cards. While this power information about the signal is quite limited, and poses several challenges in interference estimation as outlined previously, this information exposed happens to be a fundamental characteristic of every wireless technology.

Radio waves are fundamentally a form of energy (or power) emitted and absorbed by charged particles, which exhibits wave-like behavior as it travels

through space. Radio technologies then transmit messages by systematically changing (modulating) some property of these radiated waves, such as their amplitude, frequency, or phase. Thus, fundamentally, every wireless technology employs radio emissions that inherently consist of electro-magnetic energy (or power). Therefore, *a solution that is developed solely on top of such received power is generic, and is potentially applicable to all wireless technologies.*

While different radio technologies follow different protocols, employ different channel access methodologies and modulation mechanisms, they fundamentally use (electro-magnetic) transmission power to exchange information. To our advantage, the very fact that different radio technologies employ diverse mechanisms and protocols can be used to detect such devices — *diverse transmission power patterns of different radio technologies can be used to uniquely identify them*. For example, devices using Time-Division Multiplexing (TDM) will exhibit very different power patterns (in the time-frequency axis) compared to the devices using Frequency-Division Multiplexing (FDM). Similarly, frequency hoppers such as Bluetooth devices will exhibit different power patterns (*i.e.*, they emit energy in different parts of the spectrum) compared to fixed-frequency devices such as ZigBee devices. In Chapters 3 and 4, we explain our non-WiFi device detection and interference estimation procedures that make use of this fundamentally property of transmission power patterns. Our solutions use a combination of machine learning and signal clustering mechanisms to uniquely identify different non-WiFi devices based on their distinct transmission power patterns. We comment on the applicability of our work in context of future wireless LANs and newer wireless technologies in Chapter 7. Next, we formulate a precise problem statement and subsequently explain the various components involved in our overall solution.

Problem statement and solution approach

We now explain the target setting considered in this thesis along with the problem statement and our overall approach to the solution.

Target Setting: We consider the problem of identifying the reason for wireless losses and characterizing wireless interference in the context of enterprise wireless LANs. These WLANs have been deployed in a number of enterprises to provide wireless connectivity [34, 134]. Typically, an enterprise WLAN consists of a set of wireless APs that are connected through a wired backplane. A number of these deployments also follow a centralized WLAN architecture [148, 149], wherein the key configuration and management functionality is offloaded to a central control element — a WLAN controller. Typically the central WLAN controller is responsible for configuration and management of network policies, access control and other security settings, and radio resources [148]. Many wireless vendors such as Cisco [4], Aruba [10], Meru [9] have employed such central controller-based WLAN architectures.

In this thesis, we consider the target setting of such centralized enterprise WLAN architecture, and develop practical systems employing commonly available WiFi hardware to detect and quantify wireless interference (WiFi to WiFi interference as well as non-WiFi to WiFi interference) and distinguish them from losses due to weak signal scenarios. We comment on the applicability and extensibility of our proposed mechanisms in the context of other wireless environments (e.g., home wireless LANs) that do not host a central controller in Chapter 7.

Problem statement: Given a typical enterprise WLAN setting with many different APs and clients, all employing common WiFi hardware, the high level question this thesis has tried to address is the following —

> "How can we detect, quantify and localize interference from WiFi sources as well as non-WiFi sources in real-time, and all using the limited information exposed by today's commodity WiFi hardware?"

More specifically, we would like to investigate the design of systems and models that run on top of commodity WiFi hardware and enable such functionality. That is, despite the limited information provided by current wireless cards (e.g., only received power information as opposed to raw signal

information), our goal is to develop systems and models that employ such commonly available WiFi hardware and are able to (i) detect WiFi as well as non-WiFi device transmissions and uniquely identify different interferers that can be possibly of the same type (e.g., two different cordless phones of the same make and model), (ii) quantify the exact interference impact of each such WiFi or non-WiFi device, and (iii) physically pin-point the location of each such device without having information about the transmit power of this device. We now give a high level overview of the overall solution approach taken in this dissertation.

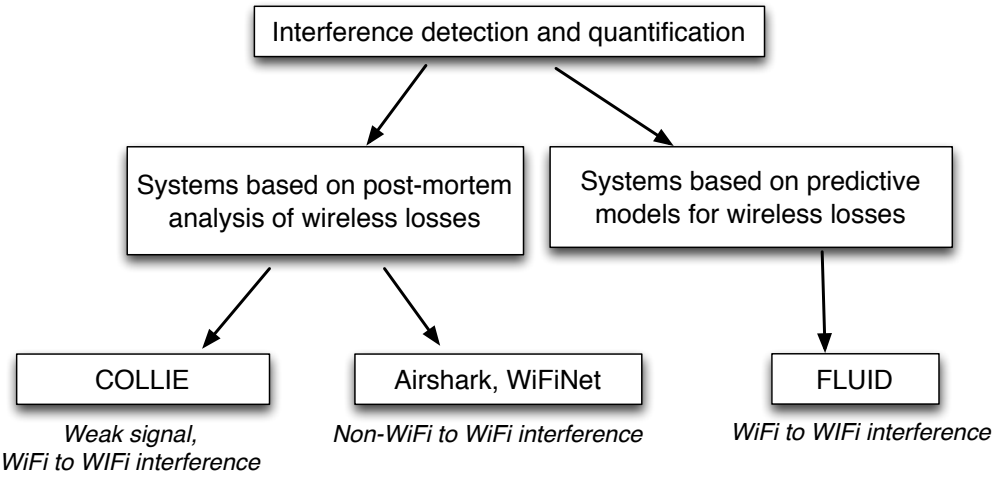

Figure 1.4: Our approach to the solution in this thesis. COLLIE, Airshark and WiFiNet are systems based on post-mortem analysis of wireless losses, whereas the FLUID system is based on a modeling approach that is predictive in nature. As shown in the figure, each of these systems can handle different sub-problems in broad space of interference detection and quantification.

Overall solution approach: We address the overall interference detection and quantification problem stated above by breaking the problem into few sub-problems and developing solutions for each of them. To begin with, we only consider WiFi to WiFi interference, and develop COLLIE (Chapter 2), a system to distinguish between WiFi packet losses that happen due to "weak signal"

and "wireless collisions" (*i.e.*, packet losses due to WiFi to WiFi interference). COLLIE uses limited information exposed by current wireless cards (e.g., RSSI and packets received in error) to accomplish this. We then focus our attention on the problem of non-WiFi to WiFi interference, and develop Airshark (Chapter 3) — a system that can detect various non-WiFi devices using WiFi-only hardware. Airshark uses received power information and learns the transmission power patterns of different non-WiFi devices to distinctly identify them in real-time. Next, we design and develop WiFiNet (Chapter 4) that builds upon Airshark, and can quantify and localize interference from various non-WiFi interfering sources. WiFiNet uses time synchronization based heuristics and clustering procedures that operate on received power information to estimate the amount of interference from each non-WiFi device. Propagation model based localization procedures that operate on this limited received power information are then developed to physically locate each device. Subsequently, we extend WiFiNet to handle both WiFi and non-WiFi interfering sources. An underlying theme of these systems is the reliance on *real-time observations* and *post-mortem analysis* of wireless losses. For example, in these systems, wireless packet losses (once they happen) are measured and correlated with other ongoing WiFi/non-WiFi device transmissions to understand the impact of these interferers. In the last part of this thesis, we design and implement FLUID (Chapter 5) that explores an alternative approach based on models that are *predictive* in nature — interaction between wireless links are modeled using signal strength based models to understand the interference relationships between these links. These models are then used to develop interference mitigation strategies. FLUID differs from our previous approaches in the sense that the interference relationships between wireless links are derived *before the event of a packet loss* using predictive models. Further, we note that FLUID is complementary to the above approaches — FLUID uses limited amount of active measurements to predict the extent of interference and possibly avoid it. Whereas, COLLIE attributes the reason for a packet loss (after it occurs) to a particular interferer or weak signal. Thus, one can envision a system that employ both COLLIE and FLUID to tackle the problem of WiFi interference — FLUID can be used to plan the transmissions of the network to avoid interference

from different WiFi links in the network (Chapter 5), whereas COLLIE can be used to detect WiFi interference that emanates in spite of such planning, due to the environmental dynamics (e.g., client mobility or new clients associating to the network). We explain each of these approaches in detail in the next few sections.

1.2 MODELING WIRELESS PACKET LOSSES DUE TO COLLISIONS

We start by studying the nature of wireless packet losses, and by developing mechanisms which help identify the cause of a wireless packet loss. In this sub-problem, we consider interference from only WiFi sources (WiFi to WiFi interference). It is well known that packet losses on a wireless link, in presence of WiFi to WiFi interference, can happen either due to a *collision* or due to a *weak signal* strength experienced at the receiver. However, discerning the exact cause of a loss, once it occurs, is known to be quite difficult. Determining the cause of a packet loss is significant as this dictates the corresponding action to be taken at the link layer — for collisions, the transmitting station would perform an exponential back-off, while for weak signal the link-adaptation algorithm controlling data-rate, transmit power parameters etc., would be invoked. Unfortunately the inability to determine the cause of a packet loss in real-time, has forced a rather conservative design for 802.11 — to start with, the cause is 'blindly' attributed to collision (thereby invoking exponential back-off) for a certain fixed number of re-transmission attempts. Further, continued failure of the re-transmissions is taken as an indication of weak signal thereby triggering rate-adaptation. Such a biased approach of assuming collision as the default cause for packet loss would translate to wasted bandwidth, energy and significant performance degradation. This is especially true in mobile usage scenarios where packet losses are more likely to occur due overly optimistic settings for data-rate/transmit power parameters rather than due to collisions.

Our focus in this sub-problem has been to develop techniques to perform loss diagnosis, specifically between collision and weak signal, which are readily implementable on contemporary 802.11 hardware. Current 802.11 hardware exposes limited information such as RSSI of the packet received

in error. Using some custom driver modifications, we were also able to retain a copy of the packet received in error. Now, comparing the original packet with the packet received in error, one can obtain the failure bit patterns — the exact bit positions that were incorrectly received. We then focussed on understanding the failure bit patterns of the received data for loss diagnosis in 802.11. Based on our analysis, we designed COLLIE - COLLision Inferencing Engine, which immediately determines the cause of a packet loss without requiring any additional transmission from the wireless client, but by using explicit feedback from the receiver. COLLIE performs intelligent analysis on received data through a combination of various metrics such as bit-level and symbol-level error patterns and received signal strength. Our design consists of two components: (i) algorithms which separate the cases of collision from weak signal through empirical analysis; (ii) a protocol which capitalizes on the judgement from the algorithms by aptly adjusting the correct link-level parameters for 802.11. Through extensive evaluations conducted on regular laptops over a wide range of experiments, we find that our collision inferencing mechanisms worked fairly well despite only limited information (e.g., RSSI, failure bit patterns) provided by current WiFi hardware. Our results show that COLLIE can provide up-to 95% accuracy in detecting packet collisions while allowing a configurable false positive rate of 2%.

We demonstrated that collision inferencing mechanisms can be used to enhance the performance of existing link adaptation mechanisms running on top of commodity WiFi hardware. As an example, we showed that collision-aware ARF (auto-rate fallback) can lead to throughput improvements between $20-60\%$ depending on the channel conditions and the amount of congestion present in the wireless environment. Through an emulation of voice calls made using Voice-over-WiFi phones, we also showed that COLLIE reduces power consumption on the client devices by decreasing the retransmission related costs by 40% for different mobility scenarios.

1.3 DETECTING NON-WIFI DEVICES USING WIFI HARDWARE

In this piece of work, we focus our attention to the problem of non-WiFi to WiFi interference. As mentioned before, a plethora of non-WiFi RF devices "share" the unlicensed spectrum along with WiFi networks. Current 802.11 systems, however, are *oblivious* to the presence of non-WiFi RF devices and hence suffer from severe performance degradation in their presence. Existing solutions require that the WLAN administrators either use sophisticated signal analyzers [3, 66], expensive software radios [98, 139] or deploy specialized hardware [12] to detect non-WiFi devices. Our focus in this work was to develop a detection mechanism that can be readily integrated in today's WLAN networks without requiring any additional sensors or hardware. To this end, we developed Airshark— a system that detects multiple non-WiFi RF devices in *real-time and using only commodity WiFi hardware*.

To motivate the need for systems such as Airshark, we first performed a detailed measurement study to characterize the prevalence of non-WiFi RF devices in typical environments — homes, offices, and various public spaces. This study was conducted for more than 600 hours over several weeks across numerous representative locations using signal analyzers [3] that establish the ground truth. We then designed and implemented Airshark, a software system that is capable of detecting non-WiFi devices using WiFi hardware. Airshark operates on top of additional data — spectrum samples, or power per sub-carrier information — provided by emerging WiFi cards such as Atheros AR 9280 to perform non-WiFi device detection. These spectrum samples provide slightly more fine-grained information about the power in the spectrum compared to the received signal strength indicator (RSSI) [133]. However, it turns out that detecting non-WiFi devices using WiFi hardware is a challenging problem despite this additional information about the spectrum provided by emerging WiFi cards for several reasons as explained previously. We also summarize these challenges in Chapter 3. The underlying reason for these challenges stems from the fact that a WiFi card is not designed to decode a non-WiFi transmission. Further, a WiFi card has to operate under a limited view of spectrum (typically 20 MHz) and in addition to the low sampling resolution

provided by these cards (Chapter 3), various other signal characteristics that are available through spectrum analyzer hardware (e.g., phase and modulation properties) are not available from today's WiFi cards.

To overcome these challenges, Airshark takes advantage of unique properties exhibited by different non-WiFi devices as observed in the power per sub-carrier samples provided by emerging WiFi cards — different radio technologies in the unlicensed band employ different physical layer and MAC layer mechanisms (e.g., modulation, channel access methods, protocols) leading to *distinct transmission power patterns* in the spectrum. Airshark learns these distinct transmission patterns of different devices and uses them as signatures to perform device detection. Specifically, Airshark operates on these power per sub-carrier samples and extracts unique features that capture the spectral and temporal properties of different non-WiFi device transmissions. These features are robust to changes in the wireless environment and are used by Airshark's light-weight decision tree classifiers to perform device detection in real-time. Airshark is able to detect multiple non-WiFi devices including fixed frequency devices (e.g., ZigBee, analog cordless phone), frequency hoppers (e.g., Bluetooth, game controllers like Xbox), and broadband interferers (e.g., microwave ovens). Airshark has an average detection accuracy of 91−96%, even in the presence of multiple simultaneously active RF devices operating at a wide range of signal strengths (−80 to −30 dBm), while maintaining a low false positive rate. Further, Airshark's performance is comparable to commercial signal analyzers that employ custom hardware. Through a deployment in two production WLANs, we also demonstrated Airshark's potential in monitoring the RF activity, and understanding performance issues that arise due to non-WiFi interference.

To the best of our knowledge, Airshark is the first system that provides a generic, scalable framework to detect non-WiFi RF devices using only commodity WiFi cards and enables non-WiFi interference detection in today's WLANs.

1.4 QUANTIFYING NON-WIFI INTERFERENCE USING WIFI HARDWARE

The previous piece of work, Airshark, can individually *detect* the presence of non-WiFi devices using WiFi hardware, however, it cannot quantify the interference impact of a non-WiFi device on a WiFi link. In this piece of work, we tackle the problem of deconstructing non-WiFi interference and quantifying their impact on WiFi links. Specifically, we design and develop WiFiNet, a system that leverages collaboration between multiple WiFi nodes running Airshark, to address both quantification of interference impact as well as localization of these interferers, as we explain below.

Quantifying non-WiFi interference impact in real-time: The mere presence of a non-WiFi device, as detected by Airshark, in the vicinity of a WiFi transmitter is not always harmful. For instance, an active analog cordless phone at a specific location, may only have a minimal impact on a particular WiFi link. We call such a low-impact non-WiFi device, a *minnow.* On the other hand, a microwave oven radiating a significant amount of energy in its vicinity might cause severe disruption to nearby WiFi links. We call such an interferer, a *whale.* We note that the impact of interference from the same non-WiFi device can quickly change over time. For instance, if the microwave oven's setting is adjusted to operate with a low power level, this device may suddenly turn into a minnow. On the other hand, if the cordless phone user moves to a different location which is closer to the WiFi link, this device might turn into a whale with respect to this WiFi link. It is even possible that the impact of the cordless phone on the WiFi link changes due to properties of the WiFi link itself. For example, when the WiFi link is operating at 54 Mbps, the disruptive impact of the cordless phone is quite high, with the impact decreasing as a rate adaptation algorithm reduces the WiFi link's PHY rate. WiFiNet tracks this continuously changing impact of non-WiFi transmitters on WiFi communication in real-time, adjusting its interference estimates immediately as operating parameters change (e.g., the microwave power setting is changed, or the WiFi device's PHY rate selection algorithm starts operating with a higher rate).

Locating non-WiFi interferers: WiFiNet also determines the physical location of such non-WiFi transmitters immediately, so that the precise source of such

interference can be determined, and if needed, such interfering devices can either be re-configured or disabled.

In sum, the unique aspects of WiFiNet are four-fold: First, WiFiNet quantifies the actual interference impact of each non-WiFi device on specific WLAN traffic in real-time, which can vary from being a *whale* — a device that currently causes a significant reduction in WiFi throughput — to being a *minnow* — a device that currently has minimal impact. WiFiNet continuously monitors changes in a device's impact that depend on many spatio-temporal factors. Second, it can accurately discern an individual device's impact in presence of multiple and simultaneously operating non-WiFi devices, even if the devices are of the exact same type. Third, it can pin-point the location of these non-WiFi interference sources in the physical space. Finally, and most importantly, WiFiNet meets all these objectives not by using sophisticated and high resolution spectrum sensors, but by using emerging off-the-shelf WiFi cards that provide coarse-grained energy samples per sub-carrier. Our deployment and evaluation of WiFiNet demonstrated its high accuracy — interference estimates were within $\pm 10\%$ of the ground truth and the median localization error was $\leqslant 4$ meters. Through these new and unique capabilities, WiFiNet provides new RF management tools for WiFi environments using off-the-shelf WiFi NICs only, obviating the need for sophisticated wireless hardware. In fact, WiFiNet can be easily implemented and integrated into enterprise WiFi APs to achieve improved mitigation strategies against non-WiFi interference for enterprise environments.

To the best of our knowledge, WiFiNet is the first system that is capable of quantifying and localizing the interference from non-WiFi devices using WiFi-only hardware.

1.5 MODELING INTERFERENCE BETWEEN LINKS USING FLEXIBLE CHANNELS

We now describe an alternative technique for understanding losses due to wireless interference. While the above pieces of work are based on *post-mortem* analysis of wireless packet losses, the conflict model [133] based approach described here is *predictive* in nature — interaction between wireless links are

modeled using signal strength based models to understand the "conflicts" (*i.e.*, carrier sensing and interference relationships) between WiFi links.

We focus on links using commodity WiFi hardware. Such hardware provides us with information about RSSI as well as packet reception status. RSSI/Signal strength based models are then built using probe measurement traffic, which helps us capture the interference relationships between different WiFi links *before the event of an actual packet loss*. These models can then be used to feed into interference mitigation mechanisms that can help avoid interference between the links of interest.

In this work, we only consider WiFi to WiFi interference and defer similar modeling of non-WiFi interference scenarios for future work (Chapter 7). Specifically, we model the interference between WiFi links that use different channel widths, and devise mechanisms to mitigate WiFi to WiFi interference using efficient channelization mechanisms as we explain below.

WiFi channels traditionally have been defined strictly to be a pre-defined center frequency and a specific channel width. It is known that interference in such fixed width systems can be modeled using signal strength based mechanisms [133]. In this work, explore the alternative method of modeling interference in the context of *flexible channels* — channels in which the center frequency and bandwidth are picked based on traffic demands, noise and interference levels across a spectral band. Such flexibility in channelization is known to be particularly useful to improve spectrum efficiency. Recently, there has been a growing attempt to explore the usefulness of flexible channels in the context of 802.11-based wireless networks [36, 110]. Current 802.11 hardware can provide a limited amount of software-level flexibility that allows transceivers to operate on such flexible channels, e.g., a fixed number of channel widths (5, 10, 20, and 40 MHz) and a permissible set of center frequencies in the 2.4 GHz or the 5.8 GHz band [67].

Using this flexibility, our focus in this work has been to develop interference models and to build an 802.11 wireless LAN employing flexible channelization that uses off-the-shelf wireless cards. In contrast to current 802.11 systems that use channels of fixed width, our proposed system, called FLUID, configures all Access Points (APs) and their clients running on top of commodity WiFi

hardware with channels defined by an appropriate choice of center frequency and width. A key property that has to be accounted for when employing variable channel widths is the following — increasing channel width for a single, isolated link potentially allows greater throughput. However, given the total transmit power used by the wireless card is a constant [36], the power per unit frequency reduces for larger widths leading to reduced SNR and poor connectivity in longer links. In FLUID, we developed modeling techniques to exploit this property of channel widths to increase transmission concurrency and improve the spatial reuse, leading to improved throughputs in a network wide setting. We showed that a main challenge in designing this system stems from unique effects of interference in presence of multiple transmitters in a network-wide setting, not explored before.

The core of FLUID consists of two key components: — (i) An efficient way to construct interference models for conflict graphs when channels can be on many different center frequencies and widths through a small number of measurements, and (ii) an interference mitigation mechanism that employs flexible channelization. We show how enhancing flexible channelization with data scheduling can maximize the number of simultaneous transmissions and result in improved throughputs. We have implemented FLUID using a central controller (assuming an enterprise WLAN setting) on a 50-node testbed using off-the-shelf Atheros wireless cards. Our results show that in our network, FLUID improves the throughput by a median of 66% compared to an approach using fixed width channels.

We note that FLUID is complementary to mechanisms such as COLLIE. FLUID uses active probe measurements to build models that capture interference relationships between links of interest prior to the event of a packet loss. Such mechanisms are predictive and preventive in nature. Whereas COLLIE's algorithms are put to use after the event of packet loss, and they help attribute the cause of a packet loss to interference or weak signal. Thus, one can potentially use both FLUID and COLLIE together in WLAN — FLUID can help determine the interference relationships between WiFi links beforehand and help avoid interference, whereas COLLIE can help reason out the losses

that occur in spite of such planning owing to the dynamics of the environment (e.g., new clients associating to the network, client mobility etc.)

Next, we present a summary of the contributions of this thesis.

1.6 CONTRIBUTIONS

Commodity WiFi cards provide us with limited flexibility and provide us with limited signal information such as RSSI and power per sub-carrier information. However, such WiFi hardware is commonly available and is an integral part of WLANs deployed today. Thus, any software solution that works on top of today's WLAN hardware would be readily deployable compared to solutions that take advantage of advanced features and flexibility provided by software radios. Keeping the constraints of these commodity WiFi cards in mind, our overall contribution of this dissertation is to enable interference interference estimation (WiFi to WiFi interference as well as non-WiFi to WiFi interference) in today's WLANs. Using commonly available WiFi hardware, we have designed and implemented models and systems that help us improve our understanding of wireless interference in today's indoor wireless environments. Specifically, the contributions of this dissertation are described as follows:

1. We designed and implemented COLLIE, the first prototype running on top of mainstream WiFi cards that can discern whether a packet loss was due to a collision or due to weak signal. While it was well known that packet loss can happen due to either of these reasons, how to determine the exact cause of a loss, once it occurs, was hitherto unknown. COLLIE discerns the cause of a loss by analyzing limited information provided by current WiFi cards. Specifically, it makes sure of RSSI information provided per packet along with the (corrupted) packet's contents. COLLIE's design consists of two components: (i) algorithms that expose statistical differences between collision and signal degradation based losses through empirical analysis; (ii) a protocol that capitalizes on the judgment from the algorithms by aptly adjusting the correct link-level parameters for 802.11. Evaluation results demonstrated that COLLIE can provide up-to 95% accuracy in

detecting collisions while allowing a configurable false positive rate of 2%.

2. We designed and implemented Airshark, the first research prototype that uses only off-the-shelf WiFi cards to detect the presence of non-WiFi wireless devices in real-time. To understand the need for systems such as Airshark, we performed the first measurement study that established the widespread usage and prevalence of non-WiFi devices across many locations [131]. Despite the challenge that WiFi cards cannot decode transmissions from non-WiFi devices, we were able to use limited signal information (power per sub-carrier information) from emerging WiFi cards and devise techniques to detect these devices. While the received power information provided by these cards is not enough to entirely reconstruct the signal, we identify a unique property that can be used to differentiate between the devices — different non-WiFi devices exhibit distinct transmission power patterns owing to the use of distinct radio technologies. Airshark learns these patterns and builds device-specific signatures using tools from machine learning. These signatures help Airshark detect multiple non-WiFi devices including fixed frequency devices (e.g., ZigBee devices, fixed-frequency analog cordless phones), frequency hoppers (e.g., Bluetooth devices, frequency hopping cordless phones, game controllers like Xbox), and broadband interferers (e.g., microwave ovens). Airshark has a detection accuracy of 91-96% and a low false positive rate ($< 0.1\%$), even in the presence of multiple simultaneously active RF devices operating at a wide range of signal strengths (-80 to -30 dBm). Through a deployment in two production WLANs, we demonstrated that Airshark can help diagnose non-WiFi interference issues.

3. We designed and implemented WiFiNet, the first system that detects, quantifies and localizes interference impact of various non-WiFi sources using commodity WiFi hardware alone. WiFiNet builds upon Airshark, and uses limited signal information (power per sub-carrier) from multiple WiFi APs to uniquely identify different non-WiFi devices even if they

are of the same type (e.g., two identical models of cordless phones). WiFiNet uses statistical clustering methods and tools from machine learning to segregate transmissions from individual non-WiFi devices. Using fine-grained timing analysis, WiFiNet also estimates the exact interference impact of each non-WiFi transmitter on every WiFi link in the WLAN, even in the presence of multiple simultaneously operating non-WiFi devices. Further, WiFiNet can physically locate the devices using WiFi-only hardware by employing novel localization mechanisms. In WiFiNet's design we also take into account carrier sensing interference, interference from WiFi sources and multiple PHY rates of operation used by WiFi links. WiFiNet's evaluation showed that interference estimates are within $\pm$10% of the ground truth and the median localization error is $\leqslant$4 meters.

4. We designed and implemented FLUID, a system that explores the alternative technique of mitigating wireless interference using models for conflict graphs [133] in WiFi networks. Traditionally, channels in typical 802.11 systems correspond to a pre-defined center frequency and a specific channel width. While it is common knowledge that using flexible channels (channels with arbitrary center frequency and width) can improve spectrum efficiency, their use in a practical setting with multiple APs and clients running real 802.11 hardware had not been explored before. Using measurements on a 50-node testbed, we showed that a key challenge in designing an 802.11 system employing flexible channelization stems from a unique effect of interference in these networks — the link conflicts (*i.e.*, carrier sensing and interference relationships) *depend* on the center frequency and channel widths used. We then designed FLUID, a system that (i) uses empirical models to capture the conflicts when channels can be on many different center frequencies and widths, (ii) improves network throughput using an interference mitigation mechanism that employs flexible channelization. FLUID's evaluation on a testbed using off-the-shelf wireless cards showed throughput improvements of up to 59%.

1.7 OUTLINE

The rest of this thesis is organized as follows. In the first part of the thesis, we develop a practical approach to detect WiFi to WiFi interference by developing mechanisms that distinguish between packet losses due to wireless collisions and those to weak signal (Chapter 2). In the second part of the thesis, we focus our attention to the problem of non-WiFi to WiFi interference (Chapters 3 and 4). We present Airshark, a mechanism to detect non-WiFi devices using WiFi-only hardware in Chapter 3. In Chapter 4, we present WiFiNet, a system that builds upon Airshark to quantify the interference impact of different non-WiFi devices and physically pin point their location. In the last part of this thesis, we explore the alternative method of understanding interference through a modeling based approach in the context of flexible channels (Chapter 5). We compare our work with alternative methods to understand and quantify interference in indoor wireless environments in Chapter 6. We conclude and discuss the avenues for further research in Chapter 7.

2 IDENTIFYING LOSSES DUE TO WIRELESS COLLISIONS

2.1 MOTIVATION

In this chapter, we describe our work on understanding and mitigating wireless interference by studying the nature of wireless packet losses. We restrict our attention to interference from WiFi devices only. We focus on non-WiFi interference in the subsequent chapters. Specifically, we try to design and develop efficient mechanisms to identify the correct cause of a packet loss in the presence of potential WiFi to WiFi interference. Fundamentally, in such a setting, wireless link losses can be caused either due to a packet collision (WiFi to WiFi interference) or due to weak signal strength at the wireless receiver. Attributing the correct cause for a packet loss is particularly important for wireless media, as the decision triggers different choice for link parameters and thus affects the overall performance of the wireless link. We call this problem of determining the accurate cause of a packet loss as collision or weak signal, as *loss diagnosis*.

Loss diagnosis in 802.11 can be challenging since by design, the receiver provides binary (i.e. whether the packet was correctly received or was lost) feedback on the reception properties of a packet. Suppose, for the purposes of our study, we had a receiver that could provide detailed diagnostic information on the reception properties of a packet. Then, could we do better than the current mechanisms used in 802.11? More systematically, we pose the following question in this chapter —

> *By analyzing the bit-level error patterns in received data and other physical layer metrics (e.g. at the symbol-level) can we determine the cause of a packet loss between collision and weak signal? Further, can we do this based on a single (or a few) packet loss(es) in real-time?*

Implications of loss diagnosis

Determining the cause of a packet loss is significant as this dictates the corresponding action to be taken at the link layer — for collisions, the transmitting station would perform an exponential backoff, while for weak

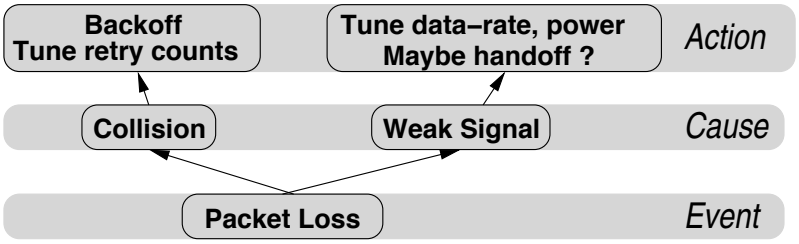

Figure 2.1: The cause of a packet loss determines which wireless link parameters have to be changed. We only consider interference from WiFi sources in the sub-problem considered in this chapter.

signal the rate-adaptation algorithm would be invoked. Figure 2.1 illustrates what must be *ideally* done in the event of a packet loss. Depending on the specific reason for packet loss, different actions should be taken at the link layer, each corresponding to adjusting different transmission parameters of the wireless interface as follows:

- **Collision:** In case of a collision related loss, the Congestion Window (CW) parameter should be double as determined by the Binary-Exponential Backoff (BEB) algorithm used in 802.11.
- **Weak signal:** For packet loss due to a weak signal, adaptation of data-rate and transmit power parameters must be performed as dictated by a specific data-rate/power adaptation algorithm.

Unfortunately the inability to determine the cause of a packet loss in real-time, has forced a rather conservative design for 802.11 — to start with, the cause is 'blindly' attributed to collision (thereby invoking exponential backoff) for a certain fixed number of re-transmission attempts. Further, continued failure of the re-transmissions is taken as an indication of weak signal thereby triggering rate-adaptation. For example, on experiencing a packet loss the transmitting station doubles the CW parameter using the BEB algorithm performs a re-transmission of the packet after appropriate backoffs (given by the new CW). If a certain number of re-transmissions fail, as determined by the tunable *Short/Long Retry Count* parameters, the station then decides to attribute the cause for packet

loss to weak signal, thereby triggering a rate/transmit power change by using appropriate rate adaptation algorithms such as Auto-rate Fallback (ARF) [80] or SampleRate [28].

Such a biased approach of assuming collision as the default cause for packet loss might be tolerable for the dominant laptop-based usage scenarios where a user is static most of time while *using* the network. However, such usage patterns are increasingly changing [64] [33] as certain emerging class of applications such Voice or Video over WiFi allow a user to be mobile while communicating with network. This creates new scenarios where constant adaptation of link parameters becomes necessary in order to operate the link at the 'best' setting. In such high mobility usage scenarios, packet losses are more likely to occur due overly optimistic settings for data-rate/transmit power parameters rather than due to collision. Therefore, the biased approach used by 802.11 could incur severe performance penalties by incorrectly attributing initial packet losses to collision.

As we move to a diverse class of applications and usage scenarios for 802.11, it is becoming increasingly important to be able to diagnose the cause of a packet loss at the link layer and trigger the correct method of adaptation in real-time. Attempts to address this problem in an indirect manner, have been observed in the design of recent approaches for rate-adaptation such as RRAA [162]. In RRAA, the station does not immediately conclude that a packet loss is due to collision or weak signal. In particular, the station performs an 'RTS test' to identify whether a certain packet loss was due to a hidden terminal, and if so, adaptively enables the RTS option to guard against future possibility of collisions from such hidden terminals. (CARA [21] also uses this approach to handle a slightly different problem.) However, the philosophy employed in RRAA and also mimicked in 802.11 is to conduct active tests or experiments (by retransmitting or sending an RRTS) to estimate collision probabilities. Being indirect, these approaches require multiple transmissions and observations to discern the channel conditions, thereby taking a long time to converge to the correct transmission parameters. In contrast, we employ a *direct* approach; we immediately determine the cause of a packet loss without requiring any additional transmissions from the wireless client,

but by conducting an empirical *post-factum* analysis of the explicit feedback obtained from the receiver.

The rest of this chapter is organized as follows. First, we present a detailed overview of our proposed solution, COLLIE (**Coll**ision **I**nferencing **E**ngine), with an emphasis on the design choices made and various components involved in the system. Next, we identify an appropriate set of metrics used for loss diagnosis through targeted experiments designed to understand collisions and a subsequent empirical analysis. Based on these metrics, we design a basic collision inferencing scheme and evaluate its accuracy through rigorous experimentation. We then propose enhancements to our basic approach using feedback from multiple APs. Finally, we modify an existing link adaptation mechanism using the COLLIE framework and evaluate its performance through experiments over various static and mobile scenarios.

2.2 AN OVERVIEW OF COLLIE

The ideas in COLLIE are motivated from the collision detection mechanism employed by the Ethernet. An Ethernet station easily detects a collision by comparing the transmitted data with the simultaneously received data. We show that, even in 802.11 systems, given a copy of the originally transmitted packet and the received error packet, it is possible to make an educated inference about the cause of transmission failure based on the error bit-patterns of this single packet. A number of different metrics are used to discern this cause, the most unique among them are the ones derived out of the constituent PHY-layer symbols of the packet. Once the cause of a packet loss is identified, this information is fed into link adaptation algorithms (such as transmit power, data rate adaptation etc.) enabling them to more intelligently select the right set of transmission parameters for all subsequent communication.

Our design (Figure 2.2) involves three components: a client module which resides on a handheld or a wireless laptop, an AP module which resides on an access point, and an *optional* backend COLLIE server which implements some additional algorithms. COLLIE places most of the optimization logic on the client device, and requires only a minimal support from the APs.

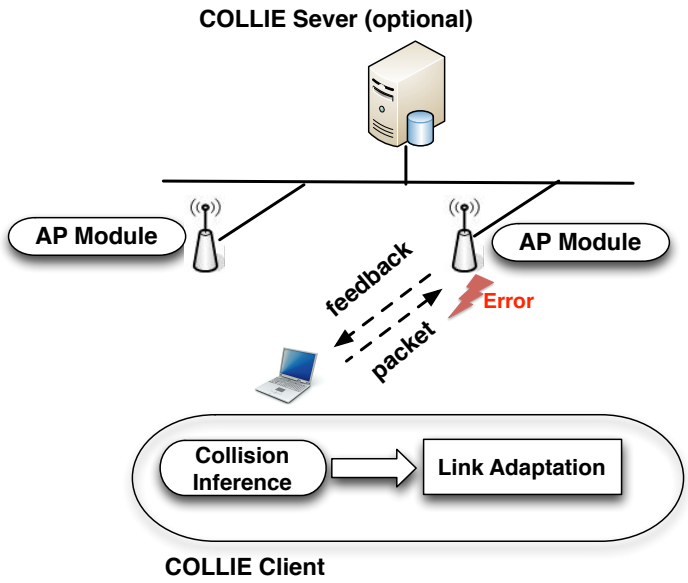

Figure 2.2: Design of our COLLIE system which consists of three modules — the client which implements a majority of the logic, the AP which performs minimal packet relaying and an optional backend server (for some specific multi-AP extensions).

Client module: The client-side COLLIE module resides at the link-layer and interacts with the link adaptation algorithms. It has access to the physical layer and MAC layer parameters and metrics such as signal strength, packet receptions, etc. Our implementation of COLLIE client module was done in a standard Linux 2.6 kernel that resides within the wireless driver as a separate kernel module. This module implements logic to discern the cause of a packet loss to either a collision or a weak signal. This process in the client is facilitated through specific feedback from the receiver, i.e., the AP, when the latter receives a packet in error. In particular, the AP relays the entire packet, received in error, back to the client for analysis. (Of course, this is only possible if the AP manages to correctly decode the source MAC address of the packet in error, which is actually quite typical.) Even though it appears wasteful, this unique and somewhat simple, type of feedback, in combination with the collision inferencing logic at the client, provides surprisingly good performance as shown by our experiments in section 2.5.

The collision inferencing algorithm analyzes the data packet that was received in error and makes an educated inference as to the cause of the packet loss. It uses a set of metrics such as received signal strength (communicated as a part of the feedback process), patterns in bit-errors and their distribution, patterns in *symbol errors* and their distribution, etc. One interesting observation in our work is that *symbol-level errors were quite useful in discerning cause of packet losses.* Section 2.3 studies this in detail through an empirical analysis.

AP module: As shown in Figure 2.2, the AP-side implementation of COLLIE includes a module, that implements the component to provide the kind of client feedback described above (and in further detail in Section 2.3). Finally, it optionally implements constructs that allow a central COLLIE server to more accurately determine the cause of a packet loss.

COLLIE server (optional): This is an optional component in our design. The COLLIE server implements a simple collision inferencing algorithm that utilizes feedback from multiple access points in the network. We show (in Section 2.3) that the accuracy of our basic collision detection mechanisms can be greatly improved by using a COLLIE server in additional to the above two modules.

2.3 FEEDBACK-BASED COLLISION INFERENCE

A critical component in COLLIE is the client side component which takes advantage of feedback from the receiver such as an AP in WLAN (or a peer if in ad-hoc mode) in order to infer the cause of a packet loss (weak signal versus collision). COLLIE implements most of the logic on the client device requiring minimal support from the receivers. We describe two versions of this inferencing algorithm. (i) A basic version (Single-AP), which requires minimal support from the AP to which the client is associated to. This applies to environments where a single AP provides wireless access to the entire establishment, such as in hotspots – coffee shops, apartments, etc. (ii) An enhanced version (Multi-AP) which builds on top of the basic version, by leveraging input from two or more APs to provide very high accuracy in detecting collisions. This approach applies to enterprise WLANs where

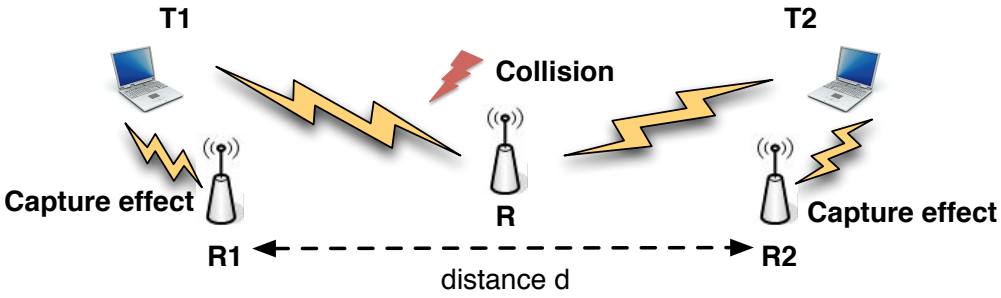

Figure 2.3: Experiment setup designed to study various metrics for inferring collisions.

multiple APs belong to the same administrative domain. As with the basic case, APs here also implement a very minimal relaying of information that assists in collision inferencing.

We evaluate our algorithm quantitatively by considering the following (i) the probability of false positives — that is, the cases where our algorithm outputs a collision while the actual cause was weak signal, and (ii) the accuracy — that is, the number of cases our algorithm identifies as collision over the total number of cases. Our design of metrics, discussed later in this section, allows the link management algorithms to specify a certain false positive rate, making the exact accuracy a function of this rate. This choice is by design, thereby leaving a significant control to the actual link management algorithms in the client. However, to provide a sense of the strong performance of our algorithms we observe that, given a desired false positive rate of 2%, our basic algorithms achieve an accuracy of about 60% on average, while the multi-AP enhacements achieve an accuracy of 95% on average.

Basic Approach (Single AP)

The basic algorithm for collision inferencing presented here, uses a simple relaying back of a data packet received in error. This relaying is done by the intended recipient of the packet which is the AP to which the client is associated to (in the infrastructure mode of 802.11). Our observations indicate that due to receiver-synchronization using the physical-layer preamble, data

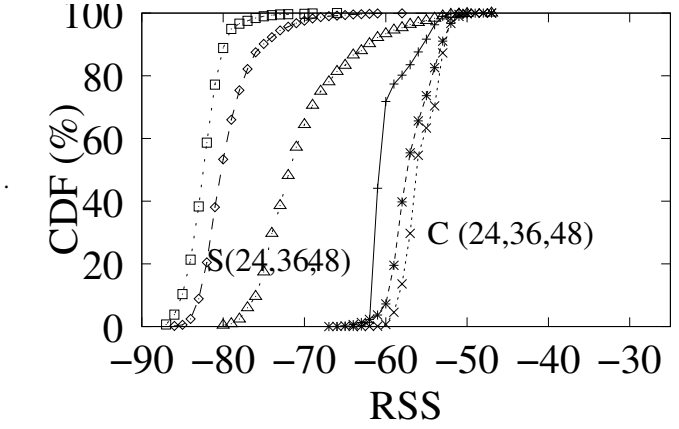

Figure 2.4: CDF of Received Signal Strength (RSS)

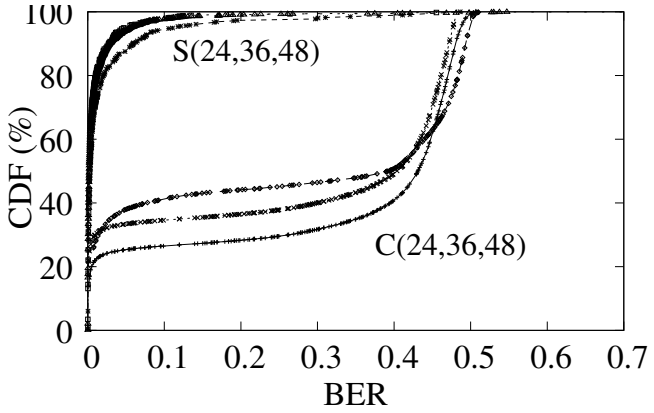

Figure 2.5: CDF of Bit-Error Rate (BER)

that immediately follows the preamble is seldom found in error — this includes critical fields in the header such as the source and destination MAC addresses. Thus, practically for all cases of packets received in error at the AP, it was possible to relay it back to the correct associated client. By analyzing these packets, we design a necessary and sufficient set of metrics comprising of bit-error rates (BER), symbol-error rates (SER), error-per-symbol (EPS), and joint distributions of these, which can act as strong indicators for packets suffering

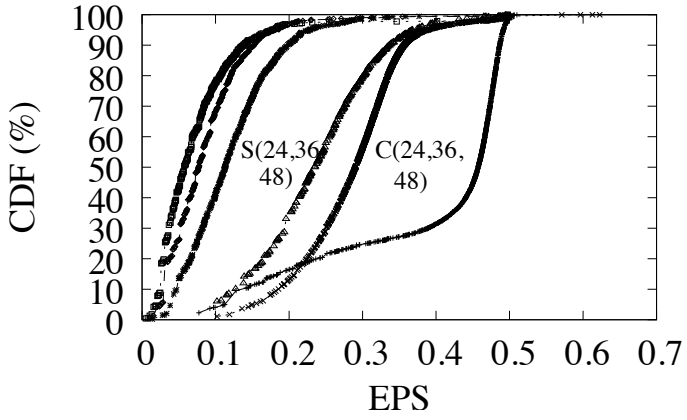

Figure 2.6: CDF of Error Rate Per Symbol (EPS)

collision versus signal attenuation. We now describe the experiments designed to understand collisions and identify the set of metrics used for loss diagnosis.

Experiment Design for Detecting Collisions

Figure 2.3 shows the experiment setup designed to induce collisions. T_1 and T_2 are two transmitters placed a certain distance apart. Receivers R_1 and R_2 are co-located with respective transmitters. Receiver R was placed in common range of both transmitters and was modified to capture and log all packets received (whether correctly or in error). The chances of collision is greatly increased by disabling the MAC-level backoffs at both T_1 and T_2. The signal between the transmitters T_1, T_2 and the receiver R was strong enough so as to not cause any bit-level errors due to attenuation. This was verified through rigorous testing. Both transmitters send broadcast packets at a fixed data-rate, thus eliminating any acknowledgments. All three receivers are opportunistically synchronized using *common* transmissions received thereby maintaining a clock skew of less than 10 μs.

To construct "ground truth," we determined the actual set of collision events by analyzing the synchronized packet logs at the receivers and identify packets that overlapped in time.

Given that we know a certain collision occurred, R observes one of the following: (1) A packet is received correctly, (2) a packet is received in error, and (3) no packet is received. Case 1 occurs when signal from one of the transmitter dominates the other resulting in a correct reception due to capture effect. Case 2 occurs when the respective signals interfere causing one of the packets to be received but with errors. Case 3 occurs when both the transmissions were perfectly synchronized, which resulted in corruption at the physical-layer header/preamble and resulting in a complete frame loss.

We performed various runs of this experiment with different data-rates and packet sizes of 1400 and 200 bytes representing long/short packets. The distance between the transmitters was set so as to sustain a certain data-rate for the broadcast packets. This ensured that no packets were received in error at R due to weak signal.

Packets in-error due to weak signal were collected using a simple process. An AP-client pair was used with unicast traffic sent from the client to the AP. Rate adaptation was enabled. Client mobility created a dynamically varying channel thereby triggering link adaptation at a packet loss. These packet losses were recorded at the AP along with additional information such as the Received Signal Strength (RSS), data-rate, etc., and used in our analysis. During the experiment, care was taken to ensure no interfering transmitters were present, thus avoiding the possibility of packet losses due to collisions.

Empirical Analysis

We present an empirical analysis of a set of metrics over the data collected through targeted experiments designed in the previous subsection.

1. Received signal strength (RSS): The received signal strength (RSS) refers to the aggregate signal plus interference $(S + I)$ measured in *dBm*. This is reported by most device drivers including the Madwifi [8] driver that we used for our experiments. The intuition behind using RSS is the following: for packets suffering a collision, their RSS is usually higher than that of packets suffering signal attenuation for the same data-rate. This observation directly follows

from the observation that packets suffering signal attenuation should have a low RSS.

Figure 2.4 plots a cumulative distribution function (CDF) for the distribution of RSS values for packets lost due to collision and weak signal. The RSS distributions are further sorted based on their data-rates; for purposes of clarity we only show data-rates of 24, 36 and 48 Mbps. In all the following plots, the legend 'C' indicates packets in error due to collision and 'S' indicates the packets in error due to weak signal. From the plot in Figure 2.4 one can observe a clear distinction in the distribution of RSS for the two categories given the same data-rate. For example, in this experiment, 98% of packets in error due to weak signal have an RSS of about -73 dBm or less, while only 10% of packets suffering collision have RSS of -73 dBm or less. Thus, by using a 'cutoff' value of -73 dBm, and it would be possible to capture about 90 % of collision cases while incurring a false-positive rate of 2%. Thus, RSS can act as a good metric for inferring the cause of packet loss.

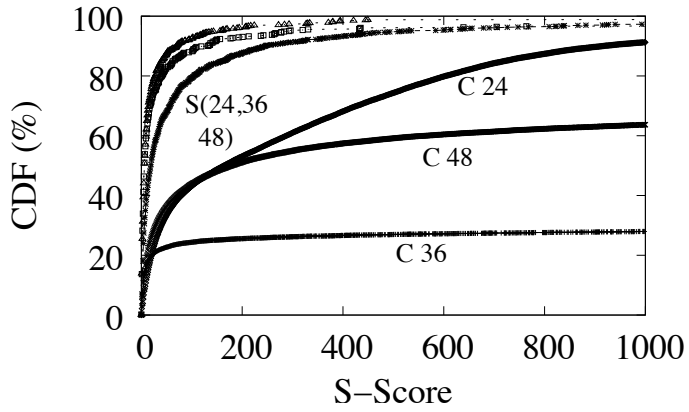

Figure 2.7: CDF of S-Score

2. Bit-error rate (BER): Much like RSS, bit-error rates (BER) for weak signal versus collision can act as a metric to distinguish with. Figure 2.5 plots the CDF of BERs for packets in error, sorted on the data-rates of 24, 36 and 48 Mbps. It follows from this plot that packets received in error due to collision have much wider distribution of BER values. For example much like RSS, 98% of packets

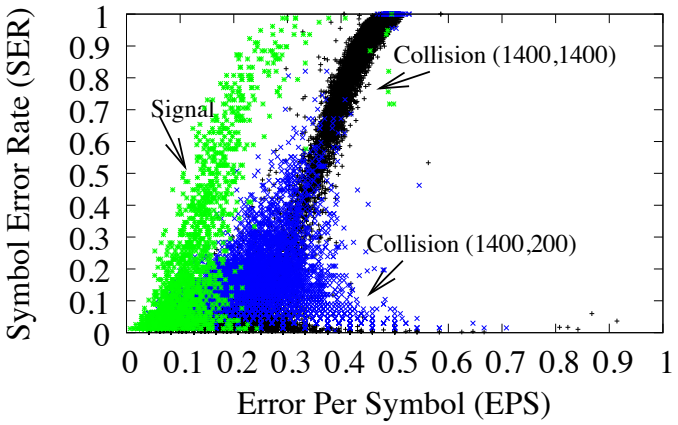

Figure 2.8: Scatter-plot of SER Vs EPS

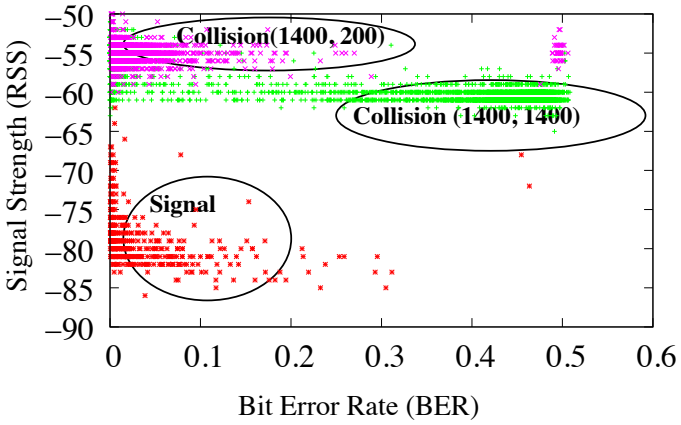

Figure 2.9: Scatter-plot of BER vs RSS.

in error due to signal have a BER of 12% or less, while only 24% of packets in error due to collision have BERs of 12% or less.

3. Metrics for capturing 'symbol-level' errors: A 'symbol' refers to a sequence of bits which are transmitted concurrently through a joint encoding and modulation method at the physical layer. For example, at 6 Mbps, the Orthogonal Frequency Division Multiplexing scheme (OFDM) uses a set of 48

sub-carriers each modulating 1 bit of information. This results in the encoding of a sequence of 48 bits in a single time-unit, which defines a symbol. Studying the patterns of symbols in error as opposed to just bits received in error can provide valuable information about the cause of a packet loss. We define a symbol to be in error if any of the bits received as a part of that symbol are in error. We studied three different metrics which exhibit certain interesting properties which we leverage in our collision inference algorithm. Note that each of these metrics are computed over every single error packet:

(i) *Symbol-error rate (SER):* Like the BER, this is the ratio of the total number of symbols received in error to the total number of symbols in the data packet. The symbol error rate indicates the actual 'amount' if error present in the packet. We have studied SER for packets in error due to collision and weak signal and we found significant overlap in its distributions. An analysis of this metric and its distributions lead us to the design of other interesting metrics which show strong results in inferring collision, described next.

(ii) *Error-per symbol (EPS):* This metric refers to the average number of bits in error among all the symbols which are in error. This is indicative of the 'amount' of error per symbol — unlike bits which have only one possible way of being in error, a 48-bit symbol received in error could have varying 'amounts' of error represented by the number of bits in error. We observe that packets in error due to collision have a larger amount of error per symbol. This is shown in Figure 2.6 which plots the CDF of EPS for both collision and weak signal. For example, 98% of packets in error due to weak signal have an EPS of 28% or less, while 45% packets in error due to collision have the same EPS of 28% or less.

(iii) *Symbol error score (S-Score):* From our study of the distributions of the symbols in error, we found that packets in collision had larger bursts of contiguous symbols in error. We designed a metric which uses 'symbol burst lengths' and computes a 'score' which we call the *S-Score* that amplifies such ambient patterns in symbol error burst lengths. We compute S-Score as $= \sum_{i=1}^{n} |B_i|^2$, where $|B_i|$ represents the length of the symbol-error bursts for burst number i. Figure 2.7 plots the CDF of the S-Score values for packets in error due to collision versus weak signal. We find that, for example, 98% of the packets in error due to weak signal have an S-Score of 500 or less, while 26%

Table 2.1: Collision Detection Accuracy and False Positive Rates

	BER	**EPS**	**S-Score**	**Metric-Vote**
Accuracy	0.550	0.524	0.441	0.597
False Positives	0.0057	0.022	0.0126	0.024

Table 2.2: Correlation Between the Metrics

	BER/EPS	**S-Score/EPS**	**BER/S-Score**
Collision	0.840	0.963	0.854
Weak Signal	0.981	0.993	0.975

packets in error due to collision have an S-Score of 500 or less. Thus, by using a cutoff of 500, we would be able to detect 74% of collision cases while incurring a false positive rate of 2%.

(iv) *Joint distribution of SER and EPS:* By considering the joint distribution of these two metrics it is possible to distinguish error packets in collision. The intuition follows from the observation that error packets in collision suffer higher symbol-error rates and correspondingly higher errors-per symbol as a function of the symbol-error rates. From the scatter plot shown in Figure 2.8, we can observe that for higher values of SER, the values of EPS get streamlined into a high yet narrow range allowing for a more accurate prediction of collision versus signal as to the cause of a packet loss.

Collision Inferencing Algorithm — Metric-Vote Scheme

Our basic collision inferencing algorithm is fairly simple. It computes the metrics discussed above on the single data packet that was received in error (relayed back by the AP). If any of the metrics indicate (vote for) a collision, the algorithm outputs collision as its inference. Even with such an aggressive approach, over the experiments performed in this section, we find that for a false positive rate of 2% (a tunable parameter), our basic approach yields a reasonable accuracy. Table 2.1 shows the results for the metrics BER, EPS, S-Score and Metric-Vote. For the cases of collision, we see that Metric-Vote has an accuracy of about 60% on an average. Later in Section 5.7, we show

that even a 60% accuracy in collision prediction can translate to significant gains in terms of throughput and energy. Next in section 2.3, we also study further enhancements to this basic scheme using support from multiple APs that can improve the accuracy to about 95% on average. For each of the metrics and the Metric-Vote scheme, Table 2.1 also shows the false positive rate — the percentage of error packets (caused due to weak signal) which the algorithms incorrectly identify as the cases of collision. We see that Metric-Vote scheme also has a low false positive rate of 2.4%. It is important to understand that the collision detection algorithms should maintain a low false positive rate. While it is beneficial to be able to decide if the packet was in error due to weak signal or collision, it would be rather costly in terms of retransmissions if we incorrectly identify a packet to be in error because of collision, when in reality it was due to a weak signal. Table 2.2 shows the correlation between the metrics – the percentage of cases where the metrics agree on their decision about the cause of the packet loss. For the cases of weak signal, the correlation between the metrics is extremely high (around 98%) evident from the fact that *all* the metrics have a very low false positive ratio. For the cases of collision, we see that the correlation drops down a little to around 85%, which improves the accuracy of Metric-Vote scheme.

Some observations: From our empirical study in the previous subsection, we found that there were a certain set of cases where inferring collision was becoming a challenge. We now explain these issues in detail:

(i) *Using RSS as a metric:* Although in general RSS acted as a good indicator of the cause of a frame loss, in some of the cases it was not able to distinguish well between the cases of collision and weak signal. This can be mainly attributed to the observed temporal variation in RSS [133]. Estimating a 'cut-off' value also becomes harder because the delivery probability is actually a function of (i) signal-to-noise ratio $S/(I + N)$ rather than $(S + I)$ which is reported by most wireless cards and (ii) receiver sensitivity [133]. However, we feel that RSS is a promising metric and could act very well when used with additional information such as RF profile of the receivers.

(ii) *Impact of physical-layer capture:* We found that there were cases of collision where the average BER for the error packet was very low due to whats known

as the *capture effect*. Capture effect refers to the phenomenon that during a collision the packet with stronger signal is received with almost no errors or a few bits in error. This experiment set up used to measure the impact of capture effect was very similar to that shown in Figure 2.3 except that now the receiver R is very close to the transmitter T_1 which resulted in a strong capture. By carefully searching for the packets received in error from T_1 (due to a collision from a concurrent transmission from T_2), we found that about 80% of packets in collision experiencing capture effect, were received with about 12% or less bits in error. This falls within our target margin of 2% false positives for the signal case thereby impacting accuracy. The accuracy of Metric-Vote scheme for strong capture effect cases was found to be around 28%.

(iii) *Effect of colliding packet size:* Using the set up in Figure 2.3, we also measured the bit error rates in collision cases for varying packet sizes. Figure 2.9 shows a scatter plot of RSS and BER for the cases of (i) weak signal (ii) collision between a 1400-byte packet and a 200-byte packet (iii) collision between two 1400-byte packets. While it is clear that using RSS in this case clearly distinguishes between the cases of collision and weak signal, using BER does not provide the same level of accuracy. In particular, we see that it becomes difficult to distinguish between cases (i) and (ii) using BER because a smaller colliding packet (200-byte in this case) would cause fewer bits in error. On the other hand, as shown in Figure 2.8, the joint distribution of SER and EPS is useful in distinguishing these cases.

Multi-AP assisted enhancements

The accuracy of our basic approach can be greatly improved if feedback from multiple APs on the packet loss could taken into consideration. This is feasible in an enterprise WLAN where APs operate in a coordinated fashion as a part of a single network. First, we present an algorithm that uses feedback from multiple APs to improve the accuracy of collision inferencing. Next, through experiments, we show that such an approach can yield good results in a practical setting.

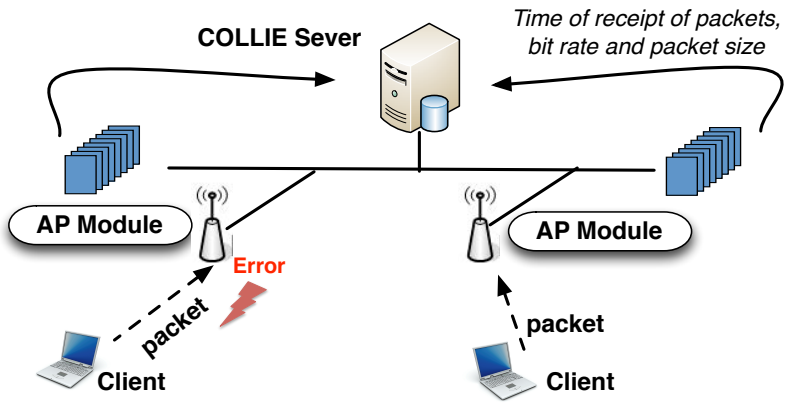

Figure 2.10: Illustration of multi-AP approach for collision inference used in COLLIE.

By leveraging feedback from two or more APs, we present an algorithm that can detect such cases and improve the accuracy of collision inferencing. Our algorithm works by aggregating such feedback at a central COLLIE server, shown earlier in Figure 2.2. The APs implement two functionalities : (i) they synchronize among each other much like the receivers R_1 and R_2 did for our experiments earlier in this section. This synchronization is done using opportunistic common packets received by the two APs on either the wired or the wireless segment. (ii) for any packet received in error, or for physical-layer error indications, the APs send a message to the COLLIE server with the time the packet (or error indication) was received, the source/destination MAC addresses and data-rate information for the packet received in error. Figure 2.10 illustrates this approach.

The COLLIE server implements a simple collision inferencing algorithm that uses time-of-receipt information about packets received in error at the APs, and combines this with information about the data-rate of the packet received to make an inference as to whether the packets did experience a collision. As a part of this algorithm the COLLIE server compares input from pairs (or a set) of APs that are known to be within range of each other. Detection of APs that are within range of each other is implemented through passive monitoring of beacons. Scenarios where APs are within each other's range are becoming fairly

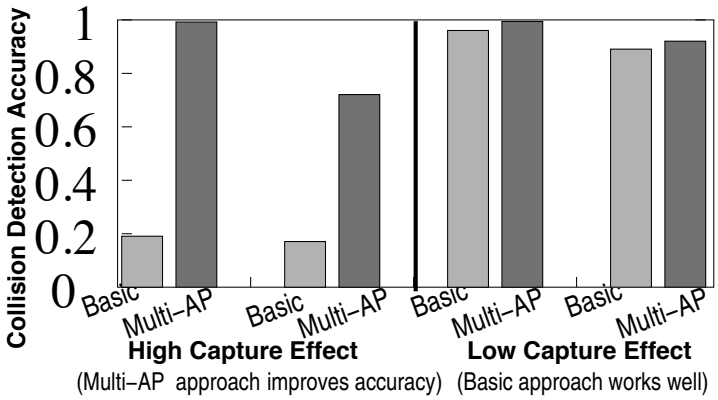

Figure 2.11: Improvements in collision detection accuracy using the Multi-AP approach.

common in todays WLANs. In fact, dense deployment of APs is promoted as an architecture for next-generation WLANs [10].

We have implemented this approach over standard Linux based APs and clients. The collision inference algorithm was implemented over a central COLLIE server. Through experiments over a simple testbed consisting of two APs and two clients we study the accuracy of our approach of using feedback from multiple APs.

Figure 2.11 shows the accuracy in collision inference using our multi-AP implementation. For the two scenarios where capture effect is dominant which were computed through experimentation within our indoor network environment, the multi-AP approach improves the accuracy of collision detection to about 95%. These two scenarios correspond to configurations where packet transmission dominate from one of the two clients respectively. For the two scenarios where capture effect is weak, both approaches provide good levels of accuracy.

2.4 USING COLLIE FOR LINK ADAPTATION

We now present a simple, yet effective protocol used to enhance link adaptation mechanisms based on the COLLIE framework. The algorithm implemented

COLLIE Module	Summary of tasks
Client	Collision inference, selective re-tx based on *Diff*
AP	Return packet in error, re-construct packet on *Diff*
Server	Facilitate multi-AP collision detection

Table 2.3: COLLIE -based link adaptation tasks in different modules

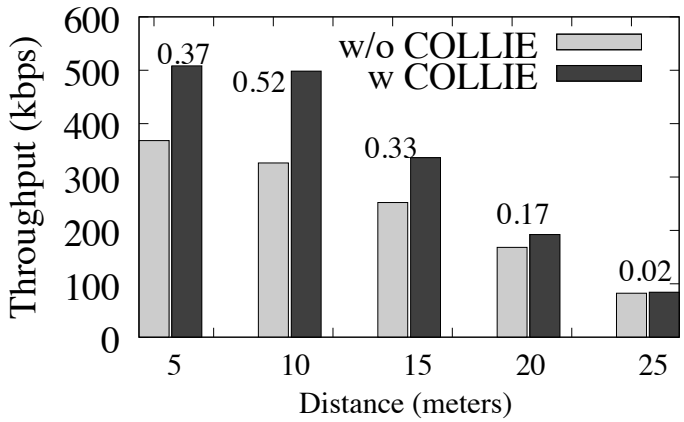

Figure 2.12: Throughput gains for static scenario

in this simple protocol is only to serve as a reference implementation of COLLIE and is by no means is an optimal algorithm. The goal of this description is to demonstrate how COLLIE can be effective in making more intelligent link adaptation decisions leading to improvements in throughput.

COLLIE-based link adaptation protocol: The goal of this link adaptation protocol is to utilize the collision inference results available from COLLIE in deciding how to best react to a packet loss and its consequent recovery. Consider a client which transmits a packet to an AP, but the latter receives the packet in error. Using feedback mechanisms, as outlined in Section 2.3 and shown in Figure 2.2, the client can infer the cause of the packet error. This knowledge is, then, fed into the link adaptation decision at the client. If the packet loss is due to a collision, then the correct adaptation mechanism is to perform exponential backoff. On the other hand, if the packet loss is determined to be

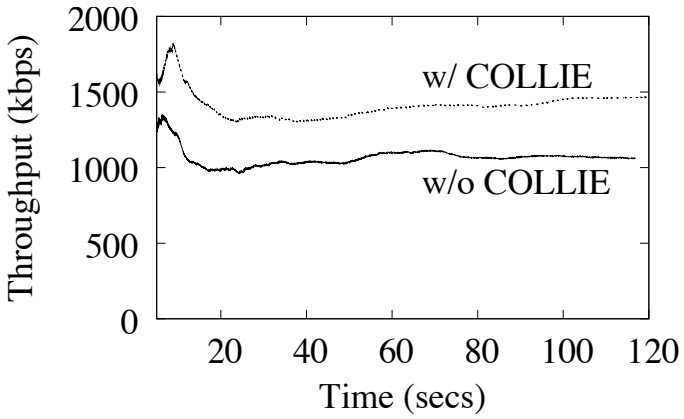

Figure 2.13: Throughput variation over time

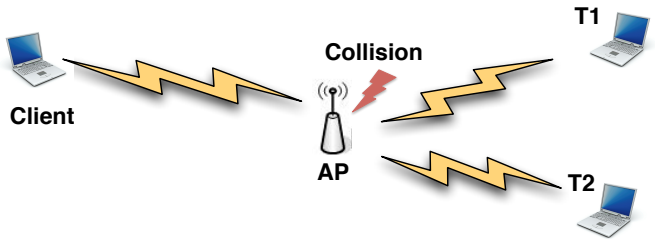

Figure 2.14: Setup for inducing collisions

due to a weak signal, then we allow an existing rate adaptation algorithm to explore and find a better data rate to transmit future packets. In general, any existing rate adaptation algorithm, e.g., RRAA, SampleRate, AARF, and ARF, can be used here to leverage such feedback from COLLIE. We explain this in the context of one of the simplest algorithm – Auto Rate Fallback (ARF). ARF uses the history of previous transmission error rates to adaptively select the data rates used for future transmissions. That is, after a number of consecutive successful transmissions, the sender attempts to transmit at a higher rate and if the delivery of this frame is unsuccessful, it immediately falls back to the previously supported mode. In our implementation, we augment the ARF algorithm with COLLIE to make it collision-aware.

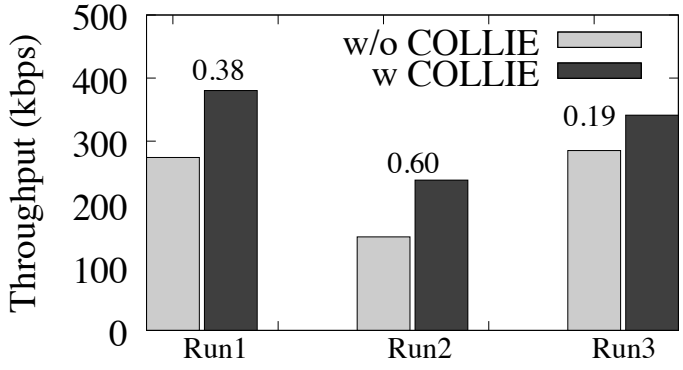

Figure 2.15: Throughput gains of COLLIE in presence of collision sources

In addition, the feedback on the erroneous packet provides another opportunity of optimization during re-transmission of a incorrectly received packet at the AP — selective re-transmission of packet segments in error. By examining the erroneous packet, the client knows exactly the set of bits that were in error. If the number of bits in error is low (say, not more than 20% of the entire packet), then it is advantageous to create a *Diff* bitmap of these bits in error and to send only this *Diff* bitmap to the AP piggybacked with the next packet transmission. If the *Diff* bitmap is correctly received, then the AP can re-construct the original packet thereby reducing the retransmission related costs associated with the client. Table 2.3 summarizes the different implementation aspects of this protocol. Note that our implementation has all the overheads due to the AP's transmission of the erroneous packet feedback, which is therefore, reflected in our performance evaluation presented next.

2.5 EXPERIMENTAL RESULTS

We now present an evaluation of COLLIE-enhanced link adaptations through experiments conducted in various static and mobile scenarios:

Experiment #1: Static scenario – Figure 2.12 shows the throughput of a static wireless client (with and without COLLIE) for increasing distance

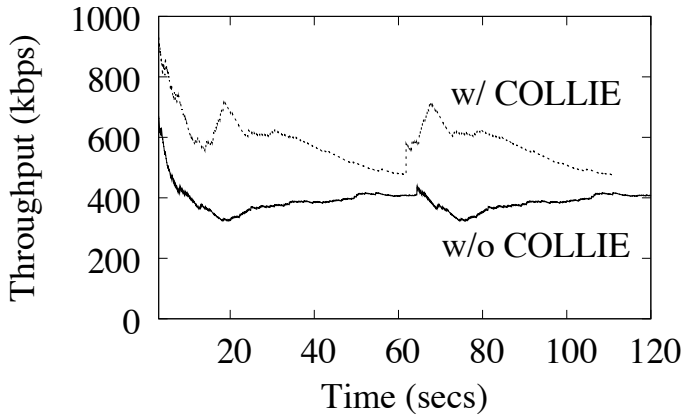

Figure 2.16: Observed throughput for mobile scenario

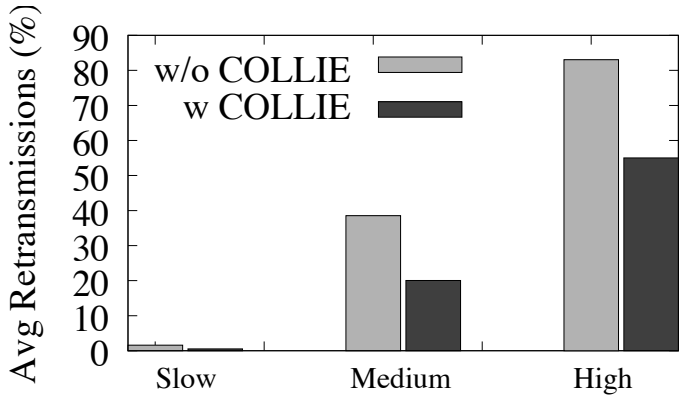

Figure 2.17: Wasted (re)-transmission as a function of channel variability induced through node mobility.

between the client and the AP. We see that as the distance between the client and AP increases, there is a corresponding drop in the throughput for both the cases. However, using COLLIE results in throughput gains of as high as 52%. On an average, we observed throughput gains of around 30%. Note that, these results account for the transmission overhead involved in the receiver feedback.

We see that after an initial increase, the throughput gains drop with the increase in distance. This is because as the channel becomes error-prone, it also becomes difficult for the AP to successfully transmit the feedback. Figure 2.12 shows that increase in throughput gains are almost negligible (2%) for these cases. Figure 2.13 plots the throughput of the client at a particular distance over time. As before, we see that using COLLIE improves the throughput by around 30%.

Experiment #2: Additional collision sources – We repeated the above experiment in presence of additional collision sources (Figure 2.14). Figure 2.15 shows the throughput improvements with and without COLLIE . We see that using COLLIE results in throughput gains of as high as 60%.

Experiment #3: Mobile scenario – For this experiment, the client position was continuously varied thereby inducing dynamic channel conditions. Figure 2.16 plots the throughput over time for both with and without COLLIE . We observe that throughput improvements using COLLIE range from around 15% to as high as 65% for the mobile scenarios. This is because COLLIE provides the rate adaptation mechanism with the information about the cause of the packet loss, thereby helping it choose the correct transmission parameters.

Experiment #4: Emulating a voice call – In this experiment, we wanted to emulate the behavior of voice traffic on the wireless medium. To do this, we made a 4 minute voice call using the Netgear SPH101 VoWiFi phone over Skype. For the duration of the call, we collected the set of packets that were sent, the time instants when they were sent, the packet sizes etc. and then replayed the exact sequence of transmissions between the wireless laptop and the access point. We conducted this experiment for low, medium and high mobility scenarios. The 'Slow' speed represents a stationary user with sporadic movement while the 'High' speed corresponds to a walking user continuously moving with a speed of about 0.5 ft/sec inside a building. Figure 2.17 shows the number of wasted 802.11 transmissions — transmissions that were not successfully received at the Access Point (AP). Under relatively high mobility conditions the percentage of wasted transmissions for 802.11 exceeded 80%. However, under the same mobility patterns, COLLIE achieves a reduction in wasted transmissions by a 40% for each of the mobility scenarios. This would

not only improve the voice quality but also result in lesser energy costs on the battery constrained mobile device.

2.6 SUMMARY OF COLLIE

In this chapter, we have tried to address the fundamental issue of identifying the cause of an erroneous packet reception in 802.11 systems. Particularly, we focussed on a setting where packet losses can be due to WiFi to WiFi interference or due to weak signal strength at the WiFi receiver. We presented the design and implementation of COLLIE, a system implementing a collision inferencing engine that analyzes the data packet received in error and makes and educated inference about the cause of the packet loss. It uses a combination of metrics such as patterns in bit and symbol error rates, their distribution etc., to infer the cause of a single packet loss — attributing it to collision (WiFi to WiFi interference) or weak signal. Unlike most of the previous approaches, our proposed mechanism, COLLIE employs a direct approach by using explicit feedback from the receiver to immediately determine the cause of the packet loss. Through rigorous evaluations conducted on regular laptops over a wide range of experiments, we find that our collision inferencing mechanisms can provide upto 95% accuracy in detecting packets in collision while allowing a configurable false positive rate of 2% and lead to throughput improvements between 20-60%. Through an emulation of voice call (made using the Netgear SPH101 Voice-over-WiFi phone), we also showed that COLLIE reduces retransmission related costs by 40% for different mobility scenarios.

3 DETECTING NON-WIFI RF DEVICES USING WIFI HARDWARE

3.1 MOTIVATION

We now focus our attention to the problem of interference from non-WiFi devices. The unlicensed wireless spectrum continues to be home for a large range of such non-WiFi devices. Examples include cordless phones, Bluetooth headsets, various types of audio and video transmitters (security cameras and baby monitors), wireless game controllers (Xbox and Wii), various ZigBee devices (e.g., for lighting and HVAC controls), even microwave ovens, and the widely deployed WiFi Access Points (APs) and clients. Numerous anecdotal studies have demonstrated that links using WiFi, which is a dominant communication technology in this spectrum, are often affected by interference from all of these different transmitters in the environment. In Figure 3.1, we present results from our own experiments where a single, good quality WiFi link was interfered by different non-WiFi devices—an analog phone, a Bluetooth device, a videocam, an Xbox controller, an audio transmitter, a frequency hopping cordless phone, a microwave, and a ZigBee transmitter—when placed at different distances from the WiFi link.

The figure shows the normalized UDP throughput under interference, relative to the un-interfered WiFi link, as a function of the interfering signal strength from these different devices. While all of these devices impede WiFi performance to a certain degree, some of these devices, e.g., the videocam and the analog phone, *can totally disrupt WiFi communication* when they are close enough ($\geqslant$ 80% degradation at RSSI $\geqslant -70$ dBm, and throughput drops to *zero* in some cases). Furthermore, our measurements across diverse home, office, and public environments, and over many weeks, show that many of these devices are routinely visible at all times of the day often at significantly high signal levels to be disruptive to WiFi links. Figure 3.2 shows an example of non-WiFi RF activity in a dorm-style apartment building, where some respite is observable only in the wee hours of the night.

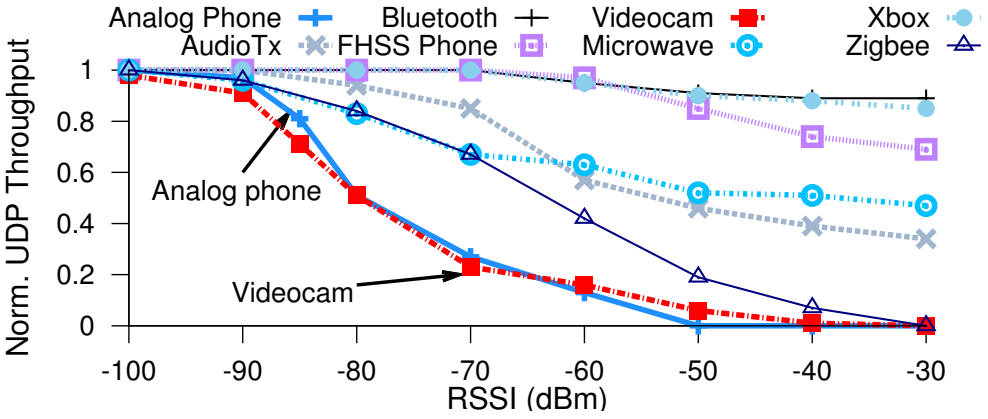

Figure 3.1: Degradation in UDP throughput of a good quality WiFi link (WiFi transmitter and receiver were placed 1m apart) in the presence of non-WiFi devices operating at different signal strengths.

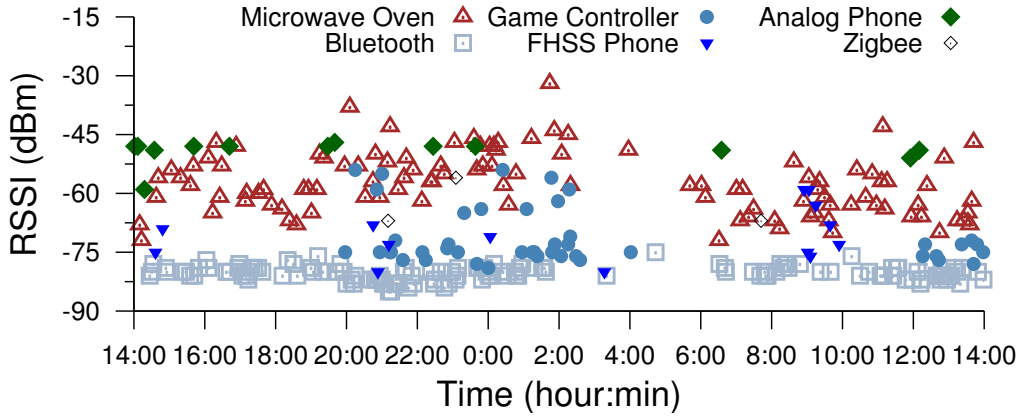

Figure 3.2: Average RSSI from different non-WiFi RF device instances shown against the device start times. Measurements were taken at a dorm-style apartment (location L16, dataset §3.2) for a 24 hour period.

Non-WiFi RF device detection: Traditional WiFi systems (Access Points or clients) utilize various mechanisms in the 802.11 MAC to detect and avoid interference from other WiFi sources, and are largely *oblivious* to non WiFi sources. However, with the continued growth of non-WiFi activity in this shared

unlicensed band and the consequent impact on WiFi performance, hardware vendors and network administrators are increasingly exploring techniques to better detect non-WiFi sources of interference. Among commercial systems, Spectrum XT [2], AirMaestro [3], and CleanAir [5] are examples of custom hardware systems that integrate unique spectrum analyzer functionality to facilitate non-WiFi device detection. In the research community, work by Hong et. al. [66] utilizes a modified channel sounder and associated cyclostationary signal analysis to detect non-WiFi devices. RFDump [98] uses USRP GNU Radios and phase/timing analysis along with protocol specific demodulators to achieve similar goals. In this chapter, we focus on techniques to detect non-WiFi RF devices but using commodity WiFi hardware instead of more sophisticated capabilities available in dedicated spectrum analyzers, expensive software radios, or any additional specialized hardware.

Using commodity WiFi hardware for non-WiFi RF device detection: Commercial spectrum analyzers or software radio platforms have specialized capability of providing "raw signal samples" for large chunks of spectrum (e.g., 80−100 MHz) at a fine-grained sampling resolution in both time domain ($O(10^5)$ to $O(10^6)$ samples per second) and frequency domain (as low as 1 kHz bandwidth). In contrast, commodity WiFi hardware, can provide similar information albeit at a much coarser granularity. For instance, Atheros 9280 AGN cards, as available to us, can only provide RSSI measurements for an individual WiFi channel (e.g., a 20 MHz channel) at a resolution bandwidth of OFDM sub-carrier spacing (e.g., 312.5 kHz), and at a relatively coarse timescale ($O(10^3)$ to $O(10^4)$ samples per second). This capability is part of the regular 802.11 frame decoding functionality. If we can design efficient non-WiFi device detection using signal measurement samples drawn from commodity WiFi hardware, then it would be easy to embed these functionalities in all WiFi APs and clients. By doing so, each such WiFi AP and client can implement appropriate mitigation mechanisms that can quickly react to presence of significant non-WiFi interference. The following are some examples.

1) A microwave oven, which typically emits high RF energy in 2.45-2.47 GHz frequencies, turns on in the neighborhood of an AP (operating on channel

11) significantly disrupting the throughput to its clients. The AP detects this microwave, infers its disruptive properties, and decides to switch to a different channel (say, 1).

2) An AP-client link experiences short term interference from an analog cordless phone in a narrowband (< 1 MHz). The AP detects the analog phone and its characteristics, and hence decides to use channel width adaptation functions to operate on a narrower, non-overlapping 10 MHz channel instead of the usual 20 MHz channel.

Summarizing, if non-WiFi device detection is implemented using only commodity WiFi hardware, both these examples are possible natively within the AP and the client without requiring any additional spectrum analyzer hardware (either as add-on boards or chipsets) to be installed in them.

Our proposed approach — Airshark

Motivated by the above examples, we propose Airshark, a system that detects non-WiFi RF devices, *using only the functionality provided by commodity WiFi hardware.* Airshark, therefore, is a software-only solution which addresses multiple goals and challenges described next.

1) Multiple, simultaneously active, RF device detection: While Airshark can most accurately detect individual non-WiFi RF devices, it is also designed to effectively discern a small number of *simultaneously operating* non-WiFi devices, while keeping false positives low.

2) Real-time and extensible detection framework: Airshark operates in real-time allowing the WiFi node to take immediate remedial steps to mitigate interference from non-WiFi devices. In addition, its detection framework is extensible—adding hitherto unknown RF device profiles requires a one-time overhead, analogous to commercial systems based on spectrum analyzers [3].

3) Operation under limited view of spectrum: Being implemented using commodity WiFi hardware, Airshark assumes that typically only 20 MHz spectrum snapshots (equal to the width of a single WiFi channel) are available for its use in each channel measurement attempt. This limitation implies that Airshark cannot continuously observe the entire behavior of many non-WiFi

frequency hoppers (e.g., Bluetooth). Further, the resolution of these samples are at least 2 orders of magnitude lower than what is available from more sophisticated spectrum analyzers [1] and channel sounders [66]. In addition to this low sampling resolution, we also observed infrequent occurances of missing samples. Finally, various signal characteristics that are available through spectrum analyzer hardware (e.g., phase and modulation properties) are not available from the commodity WiFi hardware. Therefore, Airshark needed to operate purely based on the limited energy samples available from the WiFi cards, and maintain high detection accuracy and low false positives despite these constraints.

Overview of Airshark: Airshark overcomes these challenges using several mechansisms. It uses a *dwell-sample-switch* approach to collect samples across the spectrum (§3.3). It operates using only energy samples from the WiFi card to extract a diverse set of features (§3.3) that capture the spectral and temporal properties of wireless signals. These features are robust to changes in the wireless environment and are used by Airshark's light-weight decision tree classifiers to perform device detection in real-time (§3.3). We systematically evaluate Airshark's performance in a variety of scenarios, and find its performance comparable to a state-of-the-art signal analyzer [3] that employs custom hardware.

In this chapter, we make the following contributions:

- *Characterizing prevalance of non-WiFi RF devices.* To motivate the need for systems such as Airshark, we first performed a detailed measurement study to characterize the prevalence of non-WiFi RF devices in typical environments — homes, offices, and various public spaces. This study was conducted for more than 600 hours over several weeks across numerous representative locations using signal analyzers [3] that establish the ground truth.

- *Design and implementation of Airshark to detect non-WiFi RF devices.* Airshark extracts a unique set of features using the functionality provided by a WiFi card, and accurately detects multiple RF devices (across multiple models listed in Table 3.1) while maintaining a low false positive rate

RF Device Category	Device Models (set up)	Airshark's Accuracy (low RSSI — high RSSI)
High duty, fixed frequency devices — spectral signature, duty, center frequency, bandwidth		
Analog Cordless Phones	Uniden EXP4540 Compact Cordless Phone (phone call)	97.73%—100%
Wireless Video Cameras	Pyrus Electronics Surveillance Camera (video streaming)	92.7%—99.82%
Frequency hoppers — pulse signature, timing signature, pulse spread		
Bluetooth devices (ACL/SCO)	Bluetooth-enabled devices: (i) iPhone, (ii) iPod touch, (iii) Microsoft notebook mouse 5000, (iv) Jabra bluetooth headset (data transfer/audio streaming)	91.63%—99.46%
FHSS Cordless Base/Phones	Panasonic 2.4 KX-TG2343 Cordless Base/Phones (phone call)	96.47%—100%
Wireless Audio Transmitter	GOGroove PurePlay 2.4 GHz Wireless headphones (audio streaming)	91.23%—99.37%
Wireless Game Contollers	(i) Microsoft Xbox, (ii) Nintendo Wii, (iii) Sony Playstation 3 (gaming)	91.75%—99%
Broadband interferers — timing signature, sweep detection		
Microwave Ovens (residential)	(i) Whirlpool MT4110, (ii) Daewoo KOR-630A, (iii) Sunbeam SBM7500W (heating water/food)	93.16%—99.56%
Variable duty, fixed frequency devices — spectral signature, pulse signature		
ZigBee Devices	Jennic JN5121/JN513x based devices (bulk data transfer)	96.23%—99.12%

Table 3.1: Devices tested with the current implementation of Airshark. Features used to detect the devices include: Pulse signature (duration, bandwidth, center frequency), Spectral signature, Timing signature, Duty cycle, Pulse spread and device specific features (e.g., Sweep detection for Microwave Ovens). Accurac y tests were done in presence of multiple active RF devices and RSSI values range from −80 dBm (low) to −30 dBm (high).

(§5.7). Across multiple RF environments, and in the presence of multiple RF devices operating simultaneously, average detection accuracy was 96% at moderate to high signal strengths ($\geqslant -60$ dBm). At low signal strengths (-80 dBm), accuracy was 91%. Further, Airshark's performance is comparable to commercial signal analyzers (§3.4).

- *Example uses of Airshark.* Through a deployment in two production WLANs, we demonstrate Airshark's potential in monitoring the RF activity, and understanding performance issues that arise due to non-WiFi interference.

To the best of our knowledge, Airshark is the first system that provides a generic, scalable framework to detect non-WiFi RF devices using only commodity WiFi cards and enables non-WiFi interference detection in today's WLANs.

3.2 CHARACTERIZING PREVALENCE OF NON-WIFI RF DEVICES

In this section, we aim to characterize the prevalence and usage of non-WiFi RF devices in real world networks. First, we describe our measurement equipment, and data sets.

Hardware. We use AirMaestro RF signal analyzer [3] to determine the ground truth about the prevalence of RF devices. This device uses a specialized hardware (BSP2500 RF signal analyzer IC), which generates spectral samples (FFTs) at a very high resolution (every 6 µs, with a resolution bandwidth of 156 kHz) and performs signal processing to detect and classify RF interferers accurately.

— *"Ground truth" validation.* Before using AirMaestro to understand the ground truth about the prevalence of non-WiFi devices, we benchmarked its performance in terms of (i) device detection accuracy and (ii) false positives. We activated different combinations of RF devices (up to 8 devices, listed in Table 3.1) by placing them at random locations and measuring the accuracy at

different signal strengths (up to -100 dBm). Measurements were done during late nights to avoid any external non-WiFi interference. Our results indicate an overall detection accuracy of 98.7% with no false positives. The few cases where AirMaestro failed to detect the devices occurred when the devices were operating at very low signal strengths ($\leqslant -90$ dBm).

Data sets. We collected the RF device usage measurements using the signal analyzer at 21 locations for a total of 640 hours. We broadly categorize these locations into three categories: (i) *cafes (L1-L7)*: these included coffee shops, malls, book-stores (ii) *enterprises (L8-L14)*: offices, university departments, libraries and (iii) *homes (L15-L21)*: these included apartments and independent houses. Measurements were taken over a period of 5 weeks. At some locations, we could collect data for more than 24 hours (e.g., enterprises, homes) but for others we could collect measurements only during the day times (e.g., coffee shops, malls). We now summarize our observations from this data.

Non-WiFi devices are prevalent across locations and often appear with fairly high signal strengths Figure 3.3 (top) shows the distribution of non-WiFi device instances observed per hour in different wireless environments. We observe that device instances/hr. varied across locations, with some locations showing very high device activity (e.g., a median of 22, 16 instances/hr. at locations L10 (an office), L16 (dormitory) respectively). Figure 3.3 (bottom) shows the distribution of non-WiFi device RSSI at these locations[1]. We observe that the median RSSI varied from -80 to -35 dBm. Further, for around 62% locations, we observe that 75th percentile of RSSI was greater than -60 dBm (shown using a gray line) suggesting strong non-WiFi interference.

Popularity of devices varied with locations, although few devices are popular across many locations Figure 3.4 shows the distribution of non-WiFi instances at different locations. Microwaves, FHSS cordless phones, Bluetooth

[1]Observed RSSI is dependent on the exact location where the measurement node was placed. While we tried to be unbiased, node placement in reality was influenced by few factors like availability of power connection.

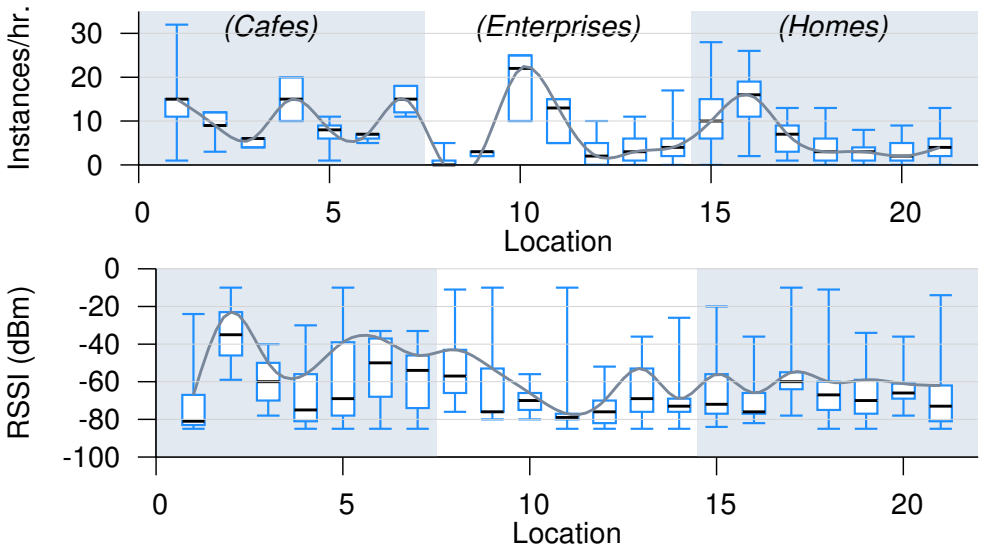

Figure 3.3: Distribution of (i) non-WiFi device instances/hour at different locations (top), and (ii) RSSI of non-WiFi devices at these locations (bottom). Min, Max, 25th, 50th and 75th percentiles are shown.

devices and game controllers were the most popular. However, some other devices appeared frequently at specific locations e.g., video cameras accounted for 29% of instances at location L4 (cafe).

Session durations for non-WiFi devices varied from a few seconds to over 100 minutes Figure 3.5 (left) shows the CDF of the session times for each class of non-WiFi devices. Many devices appear in our traces for a short duration ($\leqslant$ 2 minutes). These included (i) devices like microwaves that are activated for short durations and (ii) device instances with low signal strengths ($\leqslant -75$ dBm) that appeared intermittently at the locations where the signal analyzer was placed. However, for 25% of the cases, the devices were active for more than 5 minutes and in some traces, devices like game controllers (e.g., Xbox) were active for durations of up to 1.8 hours.

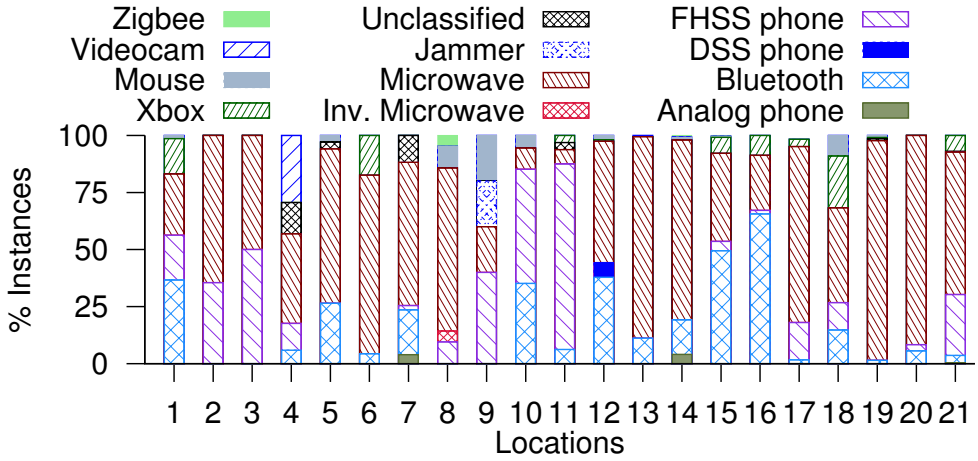

Figure 3.4: Distribution of non-WiFi device instances at various locations.

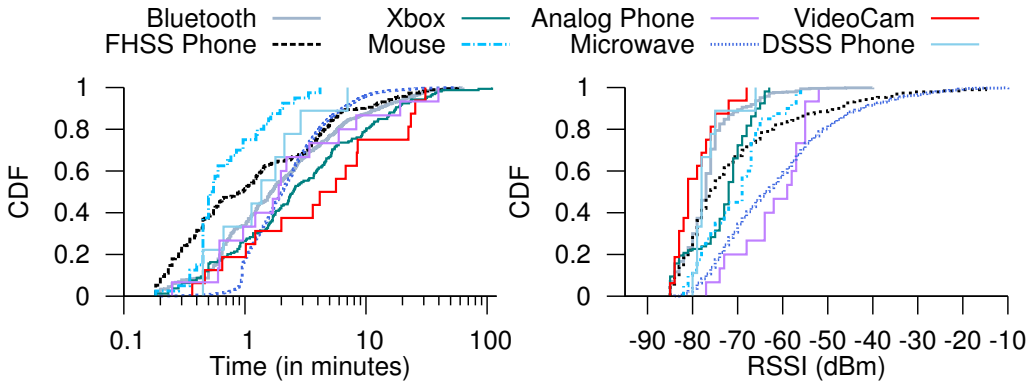

Figure 3.5: Distribution of (a) Session durations of the non-WiFi device instances (X-axis in log-scale) and (b) RSSIs of the non-WiFi device instances aggregated across all locations.

Some of the devices operate at high signal strengths indicating potential interference Microwave ovens, video cameras, and analog phones were the most dominant in terms of RSSI (Figure 3.5 (right)). For e.g., RSSI was $\geqslant -55$ dBm for more than 35% of the observed microwave oven instances, indicating

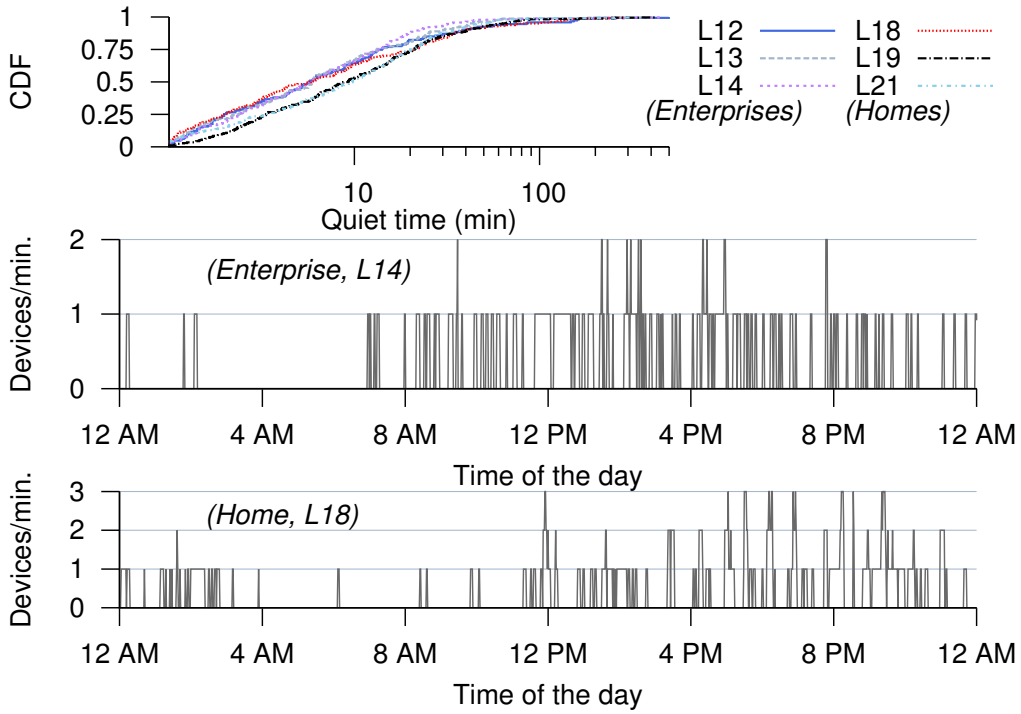

Figure 3.6: The plot shows (a) CDF of quiet times at locations where 48 hours of trace data was collected (enterprise and home locations). Time-series of non-WiFi device instances per minute at (b) an enterprise (location L14) and (c) and home (location L18) for a 24 hour period shows increased quiet times during late nights.

potential interference to nearby WiFi links. Wireless game controllers and Bluetooth devices on the other hand, mostly occurred at low to moderate RSSI of −80 to −65 dBm.

More than 50% of the periods with no non-WiFi device activity were less than 10 minutes We define a *quiet period* as a duration in which no non-WiFi devices were active. Figure 3.6 (top) shows the CDF of the quiet periods for locations (homes, enterprises) at which we collected data for 48 hours. The figure shows that non-WiFi devices appeared quite frequently—more than 50%

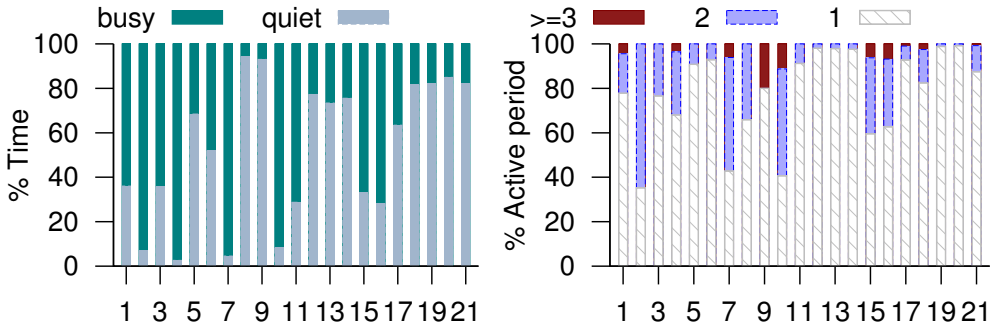

Figure 3.7: Distribution of (a) quiet and busy periods (b) simultaneously-active devices during the busy periods across different locations

of the quiet periods were less than 10 minutes, and maximum quiet period in our trace was 8 hours at L12 (office). Figure 3.6 (middle, bottom) shows a 24 hour time-series of the number of active non-WiFi devices/min. at L14 (enterprise) and L18 (home) respectively. We observe that the longer quiet periods occurred during late nights to early mornings (3 am to 8 am), and most of the non-WiFi device activity was during the daytime, when WiFi utilization is also typically significant.

At most locations, only 1 or 2 non-WiFi devices were active simultaneously for more than 95% of the busy times Figure 3.7 (left) shows that the aggregate quiet times (percentage time with no non-WiFi activity) vary at different locations. In many locations, the aggregate quiet times were around 70–80% *i.e.*, non-WiFi devices were visible for 20–30% of the time. Quiet times were much lesser for cafes (e.g., only 3% at L4) as the traces did not include the measurements during the night times and there was frequent microwave oven and cordless phone activity when we collected the traces (during the day time). Figure 3.7 (right) shows the distribution of the number of active non-WiFi devices per minute, during the busy times (periods with non-WiFi activity). We find that only a single device was active for 35% (L2, cafe) to 99% (L20, home)

of the time, and more than 2 devices were active simultaneously for at most 20% of the time (L9, office).

3.3 AIRSHARK: DEVICE DETECTION

Prior work has developed device detection mechanisms using commercial signal analyzers [3], channel sounders [66] or software radios [98] which offer fine-grained, very high resolution signal samples, typically collected using a wide-band radio. We focus on designing such a system *using commodity WiFi cards*. WiFi cards are capable of providing similar spectrum data, albeit with limited signal information, and 2 orders of magnitude lesser resolution compared to signal analyzers. Traditionally, WiFi cards have not exposed this functionality, but emerging commodity WiFi cards provide an API through which we can access *spectral samples*—information about the signal power received in each of the sub-carriers of an 802.11 channel, which opens up the possibility of detecting non-WiFi devices. Designing such a detection system using WiFi cards, however, imposes several challenges as discussed below.

Why is it hard to detect devices using WiFi cards?

- *Limited spectrum view.* Unlike sophisticated signal analyzers [1, 66] that can sample a wideband of 80−100 MHz (e.g., the entire 2.4 GHz band), current WiFi cards are designed to operate in a narrowband (e.g., 20 MHz).
- *Limited signal information.* Current WiFi cards provide limited signal information (e.g., the received power per sub-carrier) compared to software radios that provide raw signal samples. Thus traditional device detection approaches like cyclostationary analysis [66], phase analysis or use of protocol specific decoders [98] are not feasible.
- *Reduced sampling resolution.* WiFi cards have a resolution bandwidth of 312.5 kHz (equal to sub-carrier spacing) compared to signal analyzers [1] that offer resolution bandwidths as low as 1 kHz. Further, WiFi cards also have a lower sampling rate—our current implementation uses ~2.5k samples/sec, as opposed to that used by commercial signal analyzers [3]

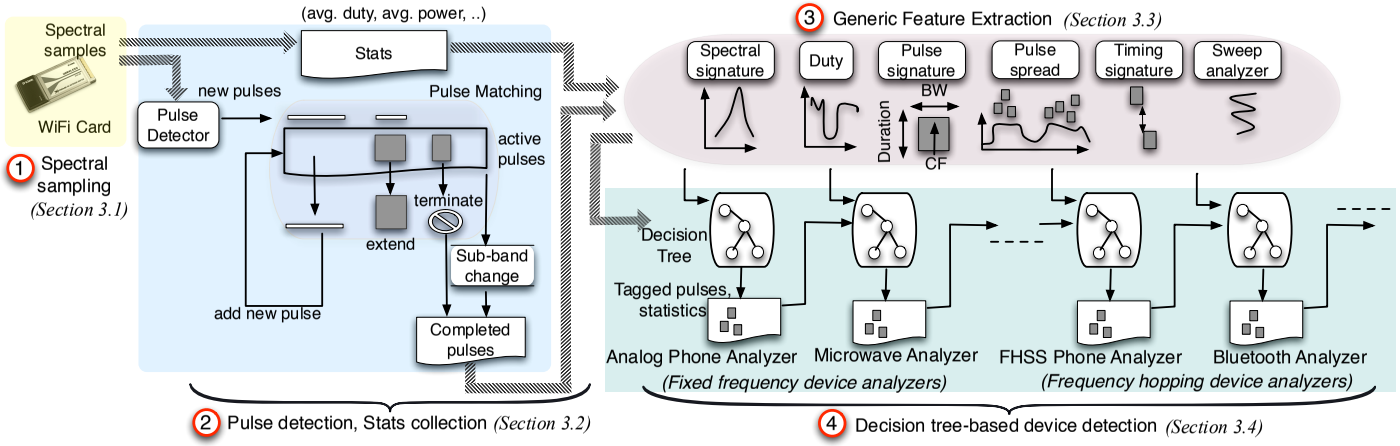

Figure 3.8: (a) Illustration of Airshark's detection pipeline. Spectral samples from the WiFi card are generated using a scanning procedure (§3.3). These samples are processed to detect signal *pulses*, and collect some aggregate statistics based on the received power values (§3.3). In the next stage, various features capturing the spectral and temporal properties of the signals are extracted (§3.3), and are used by different device analyzers that employ decision tree models (§3.3) trained to detect their target RF devices.

(160k samples/sec) or prior work [98] employing software radios (8 million samples/sec).

— *Other challenges.* Coupled with the above constraints, the presence of regular WiFi packet transmissions in the spectrum further increase the "noise" in the spectral samples, making it more challenging to detect devices.

Overview. We wish to detect the presence of multiple (simultaneously operating) non-WiFi devices in real-time. The above challenges imply that Airshark is constrained to use the *limited signal information* (spectral samples) provided by a WiFi card and must employ *light-weight* detection mechanisms that are *robust to missing samples*. We now present an overview of Airshark's device detection pipeline. Figure 3.8 illustrates the four steps listed below.

1. Spectral samples from the WiFi card are generated using a scanning procedure (§3.3). This procedure divides the entire spectrum into a number of *sub-bands* and generates spectral samples for each sub-band. Samples that comprise only WiFi transmissions are "purged" and remaining samples are passed to the next stage.
2. Next, spectral samples are processed to detect signal *pulses*—time-frequency blocks that contain potential signals of interest, and collect some aggregate statistics based on received power values (§3.3).
3. In this stage, we extract a set of *light-weight* and unique features from the pulses and statistics (§3.3)—derived using only received power values—that capture the spectral and temporal properties of signals. Example features include: *spectral signatures* that characterize the shape of the signal's power distribution across its frequency band, *inter-pulse timing signatures* that measure the time difference between the pulses, and device specific features like *sweep detection* (used to detect microwave ovens).
4. In the final stage of the pipeline, the above features are used by different device analyzers that employ decision tree models (§3.3) trained to detect their target device.

We now explain the above detection procedure in detail.

Delay	minimum	25th pc.	median	75th pc	maximum
Inter-sample time	116µs	116µs	122µs	147µs	4.94 ms
Sub-band switching	12.2 ms	14.5 ms	19.7 ms	31 ms	163 ms

Table 3.2: Time between valid consecutive spectral samples and time taken to switch sub-bands in our current implementation.

Spectral Sampling

We start by explaining the details of sampling procedure employed in our current implementation.

Spectral samples. We implement Airshark using an Atheros AR9280 AGN wireless card. We use the card in 802.11n 20 MHz HT mode, where a 20 MHz channel is divided into 64 sub-carriers, spaced 312.5 KHz apart and the signal data is transmitted on 56 of these sub-carriers. Each spectral sample (FFT) generated by the wireless card comprises the power received in 56 sub-carriers (FFT bins) and corresponds to a 17.5 MHz (56×0.3125 MHz) chunk of spectrum, which we refer to as a *sub-band*. Additionally, the wireless card also provides the timestamp t (in µs) at which the sample was taken, and the noise floor at that instant.

Purging WiFi spectral samples. We efficiently filter the spectral samples that comprise only WiFi transmissions as follows: all the samples for which Airshark's radio is able to successfully decode a WiFi packet are marked as potential candidates for purging. Airshark then reports a spectral sample for further processing only if it detects non Wi-Fi energy in that sample. To be more precise, if the radio is receiving a packet, Airshark will not report the sample unless the interference signal is stronger than the 802.11 signal being received. One downside to this approach is that Airshark will also report spectral samples corresponding to weak 802.11 signals that fail carrier detection. However, as we show in §3.3, this is not a problem as Airshark can filter out the samples relevant to non-WiFi transmissions by employing device detection mechanisms. We

term the samples reported by Airshark after this purging step as *valid* spectral samples.

Scanning procedure. Airshark divides the entire spectrum (e.g., 80 MHz) into several (possibly overlapping) sub-bands, and samples one sub-band at a time. Our current implementation uses 7 sub-bands with center frequencies corresponding to the WiFi channels 1, 3, 6, 9, 11, 13 and 14. Table 3.2 shows (i) inter-sample time: the time between two consecutive valid spectral samples (within a sub-band) and (ii) time taken to switch the sub-bands. Increased gap in the inter-sample time for a few samples ($\geqslant 150\mu s$) is due to the nature of the wireless environment—in the absence of strong non-WiFi devices transmissions, intermittent interference from WiFi transmissions causes gaps due to purged spectral samples. Sampling gaps are also caused when switching sub-bands ($\sim$ 20 ms on an average, and 163 ms in the worst case).

To amoritize the cost of switching sub-bands, Airshark employs a *dwell-sample-switch* approach to sampling: Airshark dwells for 100 ms in each sub-band, captures the spectral samples and then switches to the next sub-band. As we show later, in spite of the increased gap for few samples, we find the sampling resolution of current WiFi cards to be adequate in detecting devices (across different wireless environments) with a reasonable accuracy (§5.7). In §5.7, we demonstrate the adversarial case where strong WiFi interference coupled with weak non-WiFi signal transmissions can affect Airshark's detection capabilities.

Extracting signal data

We now explain the next stage in the detection pipeline that operates on the spectral samples to generate signal *pulses*, along with some aggregate statistics.

"Pulse" Detection. Each spectral sample is processed to identify the signal "peaks". Several complex mechanisms have been proposed for peak detection [44, 62]. To keep our implementation efficient, we use a simple and a fairly standard algorithm [49, 56]—peaks are identified by searching for "local maximas" that are above a minimum energy threshold γ_s. For each peak, the

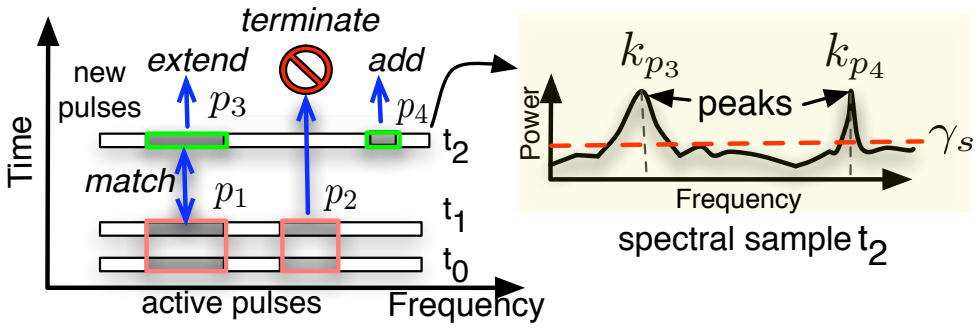

Figure 3.9: Illustration of the pulse detection and matching procedure. Pulse detector processes the spectral sample at time t_2 to output two new pulses p_3 and p_4. New pulse p_3 matches with the active pulse p_1, and results in extending p_1. Active pulse p_2 is terminated as there is no matching new pulse, and new pulse p_4 is added to the active pulse list.

pulse detector generates a *pulse* as a set of contiguous FFT bins that surround this peak. A pulse corresponds to a signal of interest, and its start and end frequencies are computed as explained below.

—*frequency and bandwidth estimation:* Let k_p denote the peak bin and $p(k_p)$ denote the power received in this bin. We first find the set of contiguous FFT bins $[k'_s, k'_e]$ such that $k'_s \leqslant k_p \leqslant k'_e$ and power received in each bin is (i) above the energy threshold, γ_s and (ii) within δ_B of the peak power $p(k_p)$ *i.e.*, $p(k) \geqslant \gamma_s \bigwedge p(k_p) - p(k) < \delta_B \forall k \in [k'_s, k'_e]$. The center frequency (CF) and the bandwidth (BW) of a pulse corresponding to this peak bin can be characterized by considering its mean localization and dispersion in the frequency domain:

$$k_c = \frac{1}{\sum_k p(k)} \sum_k k \cdot p(k), \;\; k'_s \leqslant k \leqslant k'_e$$

$$B = 2 \sqrt{\frac{1}{\sum_k p(k)} \sum_k (k - k_c)^2 \cdot p(k)}, \;\; k'_s \leqslant k \leqslant k'_e$$

The center frequency bin k_c is computed as the center point of the power distribution, and the frequency spread around this center point is defined as

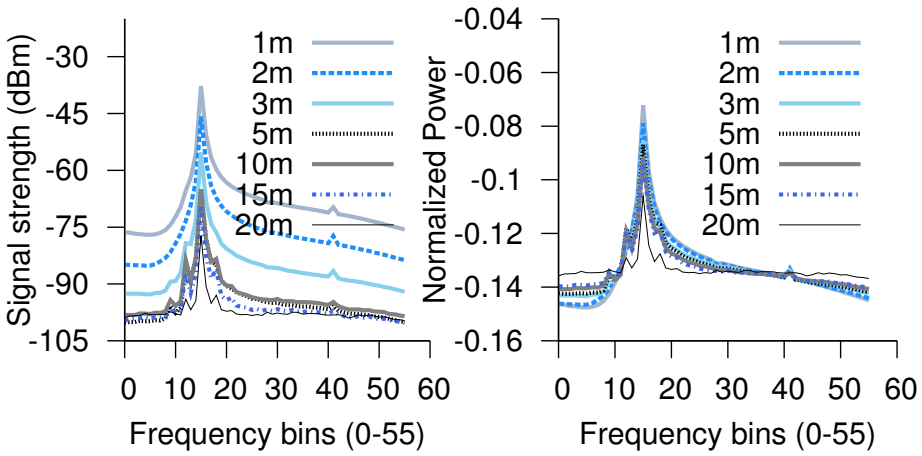

Figure 3.10: (a) Distribution of average power vs. frequency for an analog cordless phone at different distances. (b) Spectral signatures for the analog cordless phone are not affected for RSSI values of $\geqslant -80$ dBm.

the bandwidth B of the pulse. For each peak, we restrict the bandwidth of interest to comprise bins whose power values are more than γ_s and are within δ_B of the peak power $p(k_p)$. We use this mechanism as it is simple to compute and it provides reasonable estimates as we show in §5.7. Based on the computed bandwidth, the start bin (k_s) and the end bin (k_e) are determined. The pulse detector can potentially output multiple pulses for a spectral sample. Each pulse in a spectral sample observed at time t can be represented using the tuple $[t, k_s, k_c, k_e, [p(k_s) \ldots p(k_e)]]$

Pulse Matching. Airshark maintains a list of *active pulses* for the current sub-band. This active pulse list is empty at the start of the sub-band, and the first set of pulses (obtained after processing a spectral sample in the sub-band) are added to this list as active pulses. For the rest of the samples in the sub-band, a pulse matching procedure is employed: the pulse detector outputs a set of *new pulses* after processing the sample. These new pulses are compared against the list of active pulses to determine a match. In our current implementation, we use a strict criteria to determine a match between a new pulse and an active

pulse: the CFs and BWs of the new pulse and the active pulse must be equal, and their peak power values must be within 3 dB (to accommodate signal strength variations). Once a match is determined, the new pulse is merged with the active pulse to *extend* it *i.e.*, the duration of the active pulse is increased to accommodate this new pulse, and the power values of the active pulse are updated by taking a weighted average of power values of the new and the active pulse.

After the pulse matching procedure, any left over new pulses in the current spectral sample are added to the active pulse list. The active pulses that did not find a matching new pulse in the current sample are terminated. Active pulses are also terminated if Airshark encounters more than one missing spectral sample (*i.e.*, inter-sample time $\geqslant 150$ µs). Once an active pulse is terminated, it is moved to the current sub-band's list of *completed pulses*. It is possible that some of the active pulses are prematurely terminated due to the strict match and termination criteria. However, doing so helps Airshark maintain a low false positive rate as it only operates on well-formed pulses that satisfy this strict criteria (§5.7). Figure 3.9 illustrates this pulse detection procedure.

Stats Module. The stats module operates independently of the above pulse logic. It processes all the spectral samples of a sub-band to generate the following statistics: (i) *average power*: this is the average power in each FFT bin for the duration of the sub-band, (ii) *average duty*: this is the average duty cycle for each bin in the sub-band. The duty cycle of an FFT bin k is computed as 1 if $p(k) \geqslant \gamma_s$, otherwise it is 0. (iii) *high duty zones*: After processing a sub-band, a mechanism similar to peak detection, followed by CF and BW estimation procedure is applied on the "average power" statistic to identify the high duty zones in the sub-band. These are used to quickly detect the presence of high duty devices.

Before switching to the next sub-band, all the active pulses for the current sub-band are terminated and pushed to the list of the sub-band's completed pulses. The list of completed pulses along with the aggregate statistics are then passed on to the next stage of the pipeline to perform feature extraction.

Feature Extraction

Using the completed pulses list and statistics, we extract a set of *generic features* that capture the spectral and temporal properties of different non-WiFi device transmissions. These features—frequency, bandwidth, spectral signature, duty cycle, pulse signature, inter-pulse timing signature, pulse spread and device specific features like sweep detection—form the building blocks of Airshark's decision tree-based device detection mechanisms[2]. We now explain these features.

(F1) Frequency and Bandwidth. Most RF devices operate using pre-defined center frequencies, and their waveforms occupy a specific bandwidth. For e.g, a ZigBee device operates on one of the pre-defined 16 channels [100], and occupies a bandwidth of 2 MHz. The center frequency and bandwidth of the pulses (and sub-band's high duty zones) are used as features in Airshark's decision tree models.

(F2) Spectral signatures. Many RF devices also exhibit certain power versus frequency characteristics. We capture this using a *spectral signature*: given a set of frequency bins $[k_s \ldots k_e]$ and corresponding power values $[p(k_s) \ldots p(k_e)]$, if we treat the frequency bins as a set of orthogonal axes, we can construct a vector $\overrightarrow{s} = p(k_s)\hat{k_s} + \ldots + p(k_e)\hat{k_e}$ that represents the power received in each of the bins. We then normalize this vector to derive a unit vector representing the spectral signature: $\hat{s} = \frac{\overrightarrow{s}}{|s|}$. Given a reference spectral signature $\hat{s}_r$ and a measured spectral signature $\hat{s}_m$, we compute the similarity between the spectral signatures as the *angular difference* (θ): $\cos^{-1}(\hat{s}_r \cdot \hat{s}_m)$. The angular difference captures the degree of alignment between the vectors, and is close to 0° when the relative composition of the vectors is similar.

Spectral signatures can be computed on the average power values of the pulses (e.g., ZigBee pulse) or on the high duty zones (e.g., for high duty devices like analog phones) to aid in device detection. Figure 3.10 shows

[2]We selected these features with the help of attribute selection algorithm available in WEKA [11], an off-the-shelf machine learning classifier.

Protocol/Device	Bandwidth	Duration	Frequency usage
WDCT Cordless Phone	0.892 KHz	700 µs	FHSS, 90 channels
Bluetooth	1 MHz	366 µs - 3 ms	FHSS, 79 channels
ZigBee	2 MHz	$<$ 5 ms	Static, 16 channels
Game controller	500 KHz	235 µs	FHSS, 40 channels

Table 3.3: Pulse signatures for different RF devices.

the power distribution of an analog cordless phone at different distances, and the corresponding spectral signatures computed at each distance. The figure shows that normalization aids in making the signatures robust to the changes in the signal strengths of the RF devices. However, at very low signal strengths ($\leqslant -90$ dBm), the spectral signatures tend to deviate and result in an increased theta, leading to false negatives (§5.7).

(F3) Duty cycle. The duty cycle $\mathcal{D}$ of a device is the fraction of time the device spends in "active" state. This can be used to identify high duty devices, e.g., analog phones and wireless video cameras have $\mathcal{D}$=1, or identify devices with characteristic duty cycles e.g., microwave ovens have $\mathcal{D}$=0.5. In reality, due to the presence of multiple devices, it is possible for the duty cycle of the bandwidth (FFT bins) used by a device to be more than its expected duty cycle. We therefore use the notion of *minimum* duty cycle $\mathcal{D}_{\mathtt{min}}$ for devices ($\mathcal{D}_{\mathtt{min}}$=0.5 for a microwave oven) as one of the features.

(F4) Pulse signatures. Along with CF and BW, the transmission durations of many devices conform to their protocol standards. For e.g., in Bluetooth, the duration of a transmission slot is 625 µs, out of which 366 µs is spent in active transmission. Similarly, WDCT cordless phones (FHSS phones) have a pulse duration of 700 µs. Table 3.3 shows these properties (frequency, bandwidth, and duration of the pulses) for different devices. Airshark combines these three properties together to define *pulse signatures* for devices that communicate using pulses (e.g., ZigBee, Bluetooth) and uses them as features in the detection

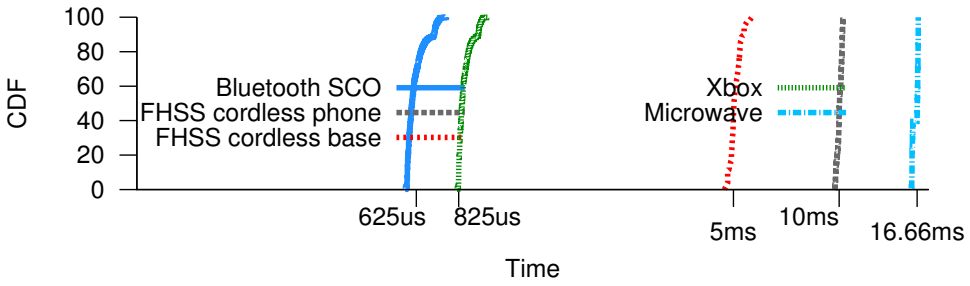

Figure 3.11: Inter-pulse timing signature for different devices.

process.

(F5) Inter-pulse timing signatures. Timing between the transmissions of many devices also exhibit certain properties. In Bluetooth SCO, for example, examining the spectrum will reveal sets of two consecutive pulses that satisfy the Bluetooth pulse signature (Table 3.3) and are separated by a time difference of 625 µs. WDCT cordless phones and game controllers (e.g., Wii) exhibit similar properties with time difference between consecutive pulses (occuring at the same center frequency) in a set being 5 ms and 825 µs respectively. Similarly, microwaves exhibit an ON-OFF cycle with a period of 16.6 ms. Figure 3.11 illustrates these timing properties.

Since Airshark can only sample a particular sub-band at a time, it cannot capture all the pulses of a device. This is especially true for frequency hopping devices. Due to the nature of sampling, we cannot expect every captured pulse to exhibit the above timing property. Airshark's device analyzers therefore use a relaxed constraint—number of pulse sets that satisfy a particular timing property is used as one of the features in the decision tree models (§3.3).

(F6) Pulse spread. Airshark accumulates the pulses for a number of sub-bands, and extracts features from pulses belonging to a particular pulse signature to detect the presence of frequency hopping devices. Together, these features represent the *pulse spread* across different sub-bands.

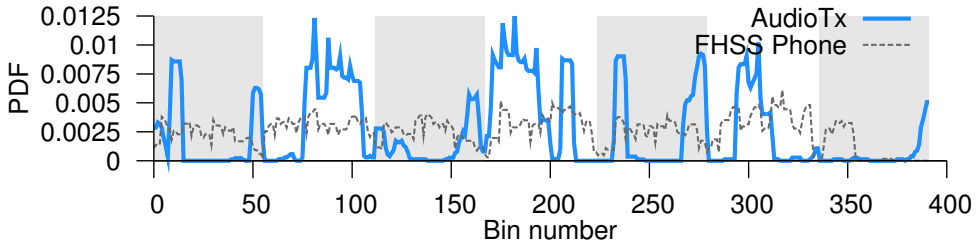

Figure 3.12: Pulse distribution of FHSS cordless phone and an audio transmitter as captured by Airshark.

1. Pulses-per-band (mean and variance). We use the average number of pulses per sub-band, and the corresponding variance as one of the measures to characterize the pulse spread. For frequency hoppers, we can expect the average number of pulses in each sub-band to be higher (and the variance lower) compared to fixed frequency devices.

2. Pulse distribution. Pulses of many frequency hopping devices tend to conform to a particular distribution. For example, FHSS cordless phone pulses are spread uniformly across the entire 80 MHz band, whereas, the pulse distribution for other frequency hoppers like audio transmitters may tend to be concentrated on certain frequencies of sub-bands, as shown in Figure 3.12. The X-axis shows the bin number b for each of the seven sub-bands ($b_{max} = 56 \times 7 = 392$), and Y-axis shows the fraction of the pulses that fall into each bin.[3]

Airshark checks whether the distribution of pulses across the sub-bands conforms to an expected pulse distribution using Normalized Kullback-Leibler Divergence (NKLD)[92], a well known metric in information theory. NKLD is simple to compute and can be used to quantify the 'distance' or the relative entropy between two probability distributions. NKLD is zero when the two distributions are identical, and a higher value of NKLD implies increased

[3]Instead of measuring the actual pulse distribution over the 80 MHz band, we stitch the sub-bands together (ignoring the overlaps) and measuring the pulse distribution over the stitched sub-bands.

distance between the two distributions. The definition of NKLD is assymetric, therefore we use a symmetric version of NKLD [92] to compare two distributions. Let $r(b)$ be the reference pulse distribution over all the bins ($b \in \mathcal{B} = [0, b_{max}]$), computed over a large period of time. Let $m(b)$ be the measured pulse distribution over a smaller time period t_m. The symmetric NKLD for two distributions $r(b)$ and $m(b)$ can be defined as:

$$\text{NKLD}(m(b), r(b)) = \frac{1}{2}\Big(\frac{D(m(b)\|r(b))}{H(m(b))} + \frac{D(r(b)\|m(b))}{H(r(b))}\Big)$$

where, $D(m(b)\|r(b))$ quantifies the divergence and is computed as $\sum_{b\in\mathcal{B}} m(b)\left|\log\frac{m(b)}{r(b)}\right|$, and $H(m(b))$ is the entropy of the random variable b with distribution $m(b)$ *i.e.*, $H(m(b)) = -\sum_{b\in\mathcal{B}} m(b)\log_2 m(b)$.

While Airshark can measure the pulse distribution $m(b)$ over a large time scale, and check if it conforms to $r(b)$, this will increase the time to detect the device. This leads to the question, "what is the minimum time scale t_m at which the pulse distribution can be measured?" We chose this time scale by empirically measuring how the NKLD values converge with the increase in the number of samples under different conditions. For the devices that we tested, we observed around 15000 samples (around 6 scans of the entire 80 MHz band, amounting to 6–7 seconds) was sufficient. We show how the number of samples affect the NKLD values in §3.4. We note that not all devices conform to a particular pulse distribution e.g., in our experiments, we found variable pulse distributions for Bluetooth as it employs adaptive hopping mechanisms.

(F7) Device specific features. Detection accuracy can be improved by using features unique to the target device. We illustrate this using a feature specific to microwave ovens.

— *Sweep detector.* The heating source in a residential microwave oven is based on a single magnetron tube that generates high power electromagnetic waves whenever the input voltage is above some threshold. This results in an ON-OFF pattern, typically periodic with a frequency of 60 Hz (frequency of the AC supply line). Although there might be differences between the emissions

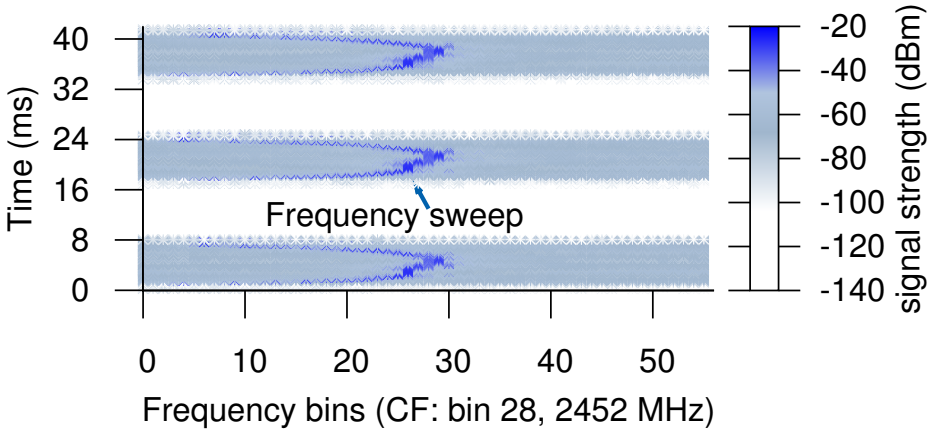

Figure 3.13: Spectral samples from Airshark capturing the activity of a residential microwave. The plot shows (i) the ON-OFF cycle for is around 16.6 ms and (ii) "frequency sweeps" during the ON periods.

from ovens of different manufacturers, the peak operational power is mostly around 2.45-2.47 GHz and during the ON periods, the radiated signal exhibits a *frequency sweep* of around 4−6 MHz [81, 150].

Figure 3.13 shows the resulting 16.66 ms periodic ON-OFF pattern and the frequency sweeps during the ON periods of a microwave oven as captured by Airshark. In the current prototype of Airshark, the microwave oven analyzer includes sweep detection, along with timing signature analysis. We tested 6 microwaves (from different manufacturers), and Airshark was able to detect all of them using these features.

Device Detection

Airshark uses decision tree [122] based classifiers in order to detect the presence of RF devices. A decision tree is a mapping from observations about an item (feature set) to conclusions about its target value (class). It employs a supervised learning model where a small set of labeled data, referred to as training data, is first used to build the tree and is later used to classify unlabeled data. In Airshark, we use the popular C4.5 algorithm [122] to construct the decision

trees. For further details about mechanisms to build decision trees and the classification process, we refer the readers to [122].

Airshark employs a separate analyzer for each class of devices. These device analyzers operate on a subset of features described previously, and make use of decision tree classifiers trained to detect their corresponding RF devices. The advantages of using per-device classifiers are three-fold: (i) each classifier can use a separate feature subset, (ii) classifiers can operate at different time granularities e.g., fixed frequency device analyzers (e.g., analog phone) can carry out the classification when Airshark finishes processing a sub-band, whereas for frequency hopping device analyzers like (e.g., Bluetooth, game controllers) the classification decision can only take place after enough samples have been processed (§3.3), (iii) classification process is more efficient when multiple devices are simultaneously active—each classifier outputs either label 1 (indicating the presence of the device), or label 0 (indicating the absence of the device). The alternative approach of using a single classifier is cumbersome as it requires training the classifier for all possible device combinations (each with a separate label).

Training. Before Airshark can identify a new RF device, its features have to be recorded for training. To do this, features relevant to this device are identified, and then extracted from spectral samples for the cases when the device is *active in isolation* (label 1), and when the device is inactive (label 0). For example, when adding the analog phone analyzer, we collected the spectral samples when the phone was activated in isolation and when the phone was inactive. We then instantiated analog phone's device analyzer to extract these features: bandwidth, spectral signature and duty cycle (measured from the recorded spectral samples) and the list of possible CFs the phone can operate on. It is worth pointing out that identifying the relevant feature set for a device and training the corresponding device analyzer is a *one time overhead before adding a new RF device* to Airshark. Table 3.1 lists the feature set employed by device analyzers in our current implementation.

Classification. We now summarize Airshark's detection pipeline. Each sample is processed by the first stage of the pipeline, and results in updating the completed pulse list and aggregate statistics. Device analyzers are invoked when Airshark finishes processing a sub-band:

1. Each device analyzer operates on the completed pulses and aggregate statistics, to derive its features. The features may include: CF, BW, angular difference (corresponding to its spectral signature), duty cycle, number and the spread of the pulses satisfying its pulse signature and timing signature.
2. The device analyzer's decision tree is invoked to output either label 1 or 0.
3. In case the decision tree outputs label 1, Airshark invokes a module that tags the selected pulses (satisfying the pulse signature and timing signature) as "owned" by this RF device.

An additional check is performed for frequency hopping device analyzers: if there are not enough accumulated samples to perform the classification, the classification decision is deferred to the next sub-band.

Dealing with multiple RF devices and overlapping signals. When multiple RF devices are simultaneously active, the spectrum may be occupied by a large number of transmissions (signal pulses). If the transmissions from multiple devices do not overlap in time or in frequency (either because of the diversity in the device transmission times, or because the devices operate in a non-overlapping spectrum bands), Airshark's device analyzers can proceed as is. Further, for certain combinations of devices, transmissions may overlap in both time and frequency, but not always. For example, this is the case when frequency hopping devices and fixed-frequency, low duty devices are present. In our benchmarks for these combinations, Airshark could always find enough pulses that do not overlap, and therefore was able to correctly detect the devices.

Transmissions from multiple devices that *always* overlap in time and frequency, however, can decrease the detection accuracy if the above techniques are used as is. For example, if the transmissions from a fixed-frequency, always-on device (e.g., analog phone) overlap in frequency with another fixed-frequency device (e.g., ZigBee device), features like spectral signatures will

not perform well. This is because overlapping signals change the "shape" of the power distribution and increase the angular difference as shown in Figure 3.14(a). One approach to resolve such overlaps, is to use cyclostationary analysis [66] on raw signal samples. Such rich signal information (very high resolution, raw signal samples), however, is not available through WiFi cards. We extend the basic approach used in Airshark to handle these cases as follows: device analyzers first identify the potential peaks that match their CFs. For each peak, instead of a complete match on the signal's bandwidth BW, a *partial match* of the spectral signatures is performed. The bandwidth $\mathrm{BW}_{\mathtt{par}}$ for the partial match is decided by the bandwidth detection algorithm, and is required to be above a minimum bandwidth $\mathrm{BW}_{\mathtt{min}}$ in order to control the false positives (*i.e.*, $\mathrm{BW}_{\mathtt{min}} \leqslant \mathrm{BW}_{\mathtt{par}} \leqslant \mathrm{BW}$). In our benchmarks, setting $\mathrm{BW}_{\mathtt{min}}$ to $0.6\times$BW improved the accuracy without increasing the false positives (§5.7). Figure 3.14(b) shows the reduction in angular difference when using a partial match.

Alternative classifiers. We also built another classifier based on support vector machines (SVM) [7]. In our experiments, we found that in most cases, Airshark's SVM-based and decision tree based classifiers had similar detection accuracies. In some scenarios involving multiple devices, SVM-based classifier performed slightly better (§3.4). However, we elected to use a decision tree based classifier, as it has very low memory and processing requirements, thus making it feasible to embed non-WiFi device detection functionality in commodity wireless APs and clients.

3.4 EXPERIMENTAL RESULTS

In this section, we evaluate Airshark's performance under a variety of scenarios, and present real-world applications of Airshark through a small scale deployment. We start by presenting the details of our implementation and testbed set up, followed by the metrics used for evaluation.

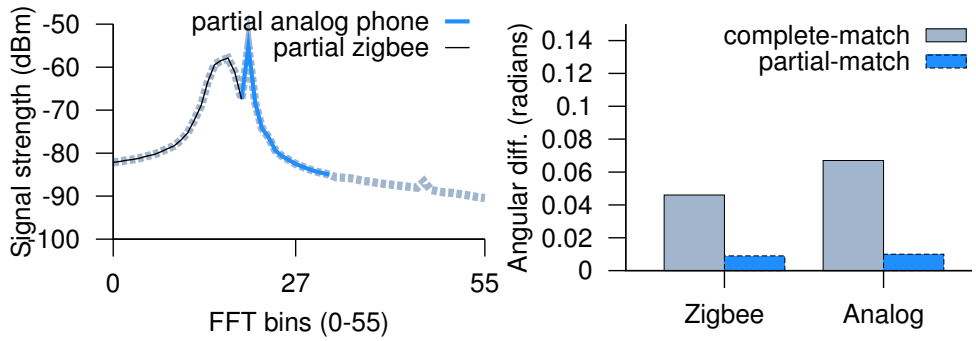

Figure 3.14: Overlapping signal detection. (a) partial overlap between ZigBee and analog signals (b) using partial matches between spectral signatures reduces the angular difference in overlapping cases.

Implementation. Our implementation of Airshark consists of few hundred lines of C code that sets up the FFT sampling from the Atheros AR9280 AGN based wireless card, and about 4500 lines of Python scripts that implement the detection pipeline. We used an off-the-shelf implementation of the C4.5 decision tree algorithm [11] for training and classification. For the alternative SVM-based classifier, we used an SVM implementation [7] employing a radial basis function kernel with default parameter setting. We focus on detecting devices in 2.4 GHz spectrum, and our current prototype has been tested with 8 classes of devices (across multiple device models) mentioned in Table 3.1.

Evaluation set up. We performed all our experiments in a university building (except those in §3.4). Our training data was taken during the late evenings and night times to minimize the impact of external non-WiFi interference. Our evaluation experiments, however, were performed over a period of one week that included both busy hours and night times. We also used the AirMaestro signal analyzer [3] in order to determine the "ground truth" about the presence of any external non-WiFi RF devices during our experiments.

Evaluation Metrics. We use the following metrics to evaluate the performance of Airshark:

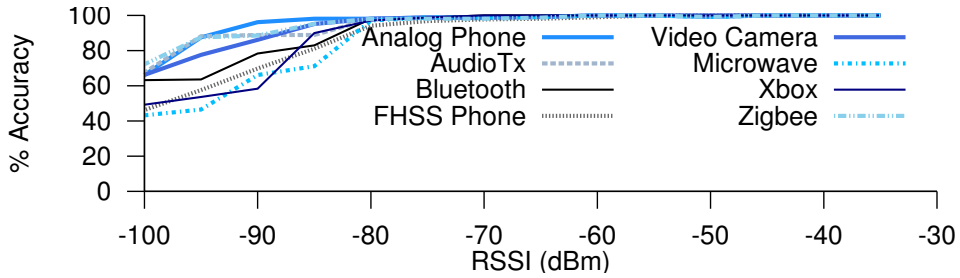

Figure 3.15: Accuracy of single device detection across signal strengths for different RF devices.

1. Detection accuracy: This is the fraction of correctly identified RF device instances. This estimates the probability that Airshark accurately detects the presence of an RF device.
2. False positive rate (FPR): This is defined as the fraction of false positives. This estimates the probability that Airshark incorrectly determines the presence of an RF device.

We will first evaluate Airshark's performance in various scenarios, and then comment on the parameters we chose. We set the energy threshold γ_s to -95 dBm, δ_B to 10 dB, and for computing NKLD we use 15000 samples. The RF devices tested and the features used are listed in Table 3.1.

Performance evaluation

We start by evaluating Airshark using controlled experiments with different RF devices.

Single device detection accuracy

Method. We measured the accuracy of device detection when only one of the RF devices mentioned in Table 3.1 was activated. The methodology used to activate the devices is also listed in Table 3.1. We placed the devices at random locations to generate the samples at different RSSI values, and then computed

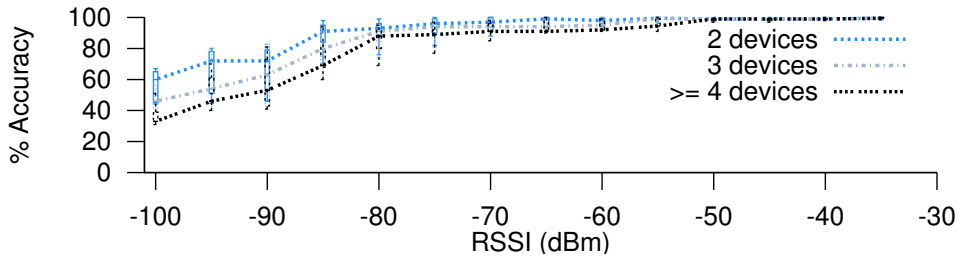

Figure 3.16: Accuracy of detection across signal strengths for 2, 3, and $\geqslant$ 4 device combinations.

the average detection accuracy at each RSSI.

Results. Figure 4.10 shows the detection accuracy as a function of RSSI for different RF devices. We observe that Airshark achieves an accuracy of 98% for RSSI values as low as -80 dBm. For RSSI values $\leqslant -80$ dBm, the accuracy drops down due to the reduced number of pulses detected at such low signal strengths. Further, the drop is sharper for frequency hopping devices, compared to fixed frequency, high duty devices like analog phones and video cameras.

Multiple device detection accuracy

Method. For each run, we chose $2 \leqslant n \leqslant 8$ random devices from our device set, placed them at random locations and activated them simultaneously to generate samples at different RSSI values. We then computed the average detection accuracy at each RSSI. We repeat the experiments for different combinations of devices and locations. We note that our experiments include the "overlapping signal" cases (§3.3).

Results. Figure 4.11 shows the detection accuracy for 2, 3, and $\geqslant$ 4 device combinations. We observe that even when $\geqslant$ 4 devices are activated simultaneously, the average detection accuracy is more than 91% for RSSI values as low as -80 dBm. For higher RSSI values ($\geqslant -60$ dBm), the detection accuracy was 96%. For lower RSSI values, in the presence of multiple RF devices, we observed that features like spectral signatures, duty cycles do not perform well,

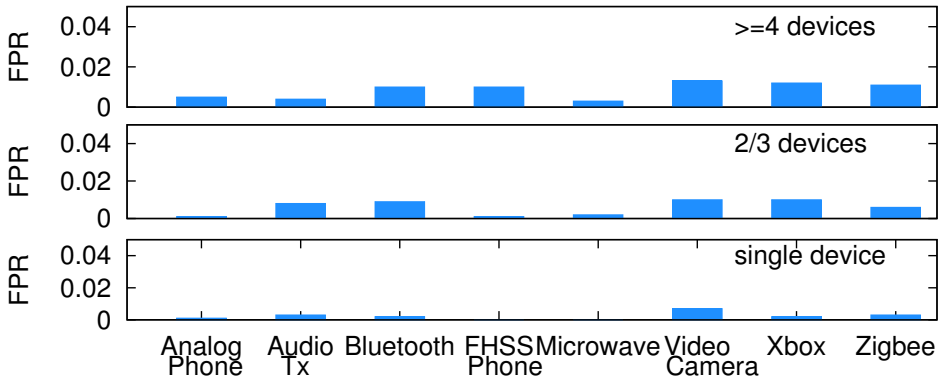

Figure 3.17: False positive rate for different devices.

and hence result in reduced accuracy (§3.4). Overall, we find that Airshark is reasonably accurate, and as we show in §3.4, its performance is close to that of signal analyzers [3] using custom hardware.

False positives

Method. When performing the above experiments for single and multiple device detection, we also recorded the false positives. Figure 3.17 shows the distribution of false positive rate across different RF devices.

Results. We observe that Airshark has a particularly low false positive rate — even when using $\geqslant 4$ RF devices, operating under a wide range of signal strengths, the average FPR was 0.39% (maximum observed FPR was 1.3%). Further, for RSSI values $\geqslant -80$ dBm, the average FPR was $\leqslant 0.068\%$.

Overall performance summary. For a total of 8 classes of RF devices used in our evaluation, across multiple runs and in presence of simultaneous activity from multiple RF devices at different signal strengths, Airshark exhibits detection accuracy of $\geqslant 91\%$ even for very low signal strengths of -80 dBm. The average false positive rate was 0.39%. At higher signal strengths ($\geqslant -60$ dBm) the accuracy was $\geqslant 96\%$.

Location/environment	Accuracy	False +ves
Indoor offices (floor-to-ceiling walls)	98.47%	0.029%
Lab environment (cubicle-style offices)	94.3%	0.067%
Apartments (dormitory-style)	96.21%	0.043%

Table 3.4: Airshark's performance in different environments.

In a typical enterprise deployment with multiple APs running Airshark, performance at low RSSI might not be a concern as we can expect at least one AP to capture the non-WiFi device signals with RSSI $\geqslant -80$ dBm. Below, we benchmark the performance under the cases with RSSI $\geqslant -80$ dBm. We revisit the performance at lower RSSI in §3.4.

Location insensitivity

Method. To understand whether the peformance of our decision tree models was affected by the location and the nature of the wireless environment, we repeated the controlled experiments in three different environments. In each case, we activated the RF devices at different signal strengths and measured Airshark's performance.

Results. Table 3.4 shows that Airshark performs reasonably well under all the three environments with an average detection accuracy of 94.3%—98.4% and an average FPR of 0.029%—0.067%. This shows that our decision tree models are general, and are applicable in different environments.

Performance of SVM-based classifier

Method. We compared the performance of SVM-based implementation of Airshark with the decision tree based version. Both SVM and decision tree implementations were trained using the same data. Similar to the previous experiments, we placed the RF devices at random locations to evaluate the performance at different signal strengths.

RF device	Airshark-SVM (%) Accuracy/FPR	Airshark-DTree (%)Accuracy/FPR
Analog cordless phone	98.31% / 0.037%	97.73% / 0.012%
Bluetooth (ACL/SCO)	92.03% / 0.094%	91.63% / 0.076%
FHSS cordless phone	98.44% / 0.052%	96.47% / 0.037%
Microwave oven	94.02% / 0.012%	93.16% / 0.06%
ZigBee device	97.49% / 0.048%	96.23% / 0.036%
Video camera	94.24% / 0.08%	92.70% / 0.072%
Audio tx/headphones	92.27% / 0.016%	91.23% / 0.014%
Game controller (Xbox/Wii)	90.32% / 0.064%	91.75% / 0.046%

Table 3.5: Comparison of SVM and decision tree based approaches. Table shows per-device accuracy in the presence of multiple RF devices. The RSSI of the devices were $\geqslant -80$ dBm.

Detection device	Online tests	Accuracy	False +ves
AirMaestro [3]	1827	1803 (98.7%)	NA
Airshark	1827	1761 (96.3%)	12 (0.07%)

Table 3.6: Comparison of Airshark and a detection device that uses a specialized hardware (AirMaestro RF signal analyzer).

Results. Table 3.5 shows that the performance of SVM and decision tree for different RF devices. We observe that while SVM based implementation performs slightly better in terms of the detection accuracy (an improvement of up to 4%), the number false positives also increase. We elected to use the decision tree approach as it was much faster and has comparable performance.

Comparison with specialized signal analyzers

Method. We compared the accuracy of Airshark with the AirMaestro RF signal analyzer [3] by employing following methodology: we performed experiments by activating a combination of RF devices at different signal strengths and collected traces from both Airshark and the AirMaestro device simultaneously.

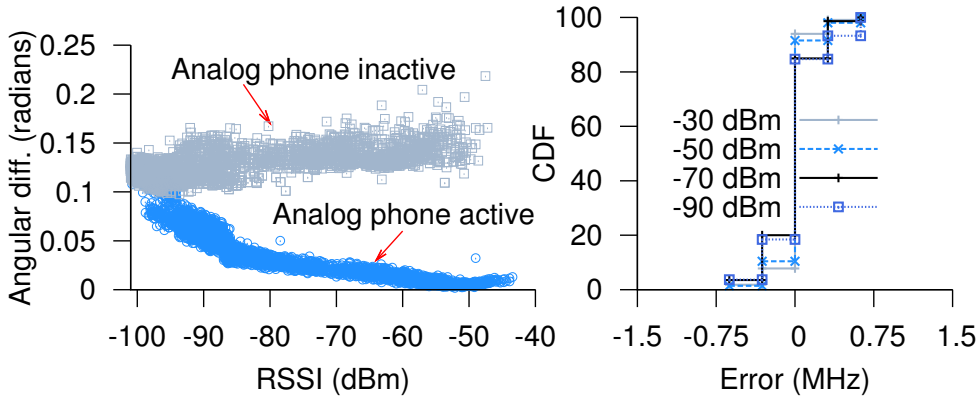

Figure 3.18: (a) RSSI vs. angular difference with respect to analog phone's spectral signature when the device is switched on and off (b) CDF of bandwidth estimation error at different signal strengths.

Table 3.6 shows the results.

Results. We observe that out of 1827 device instances, AirMaestro was correctly able to detect 1803 (98.7%), whereas Airshark detected 1761 (96.3%) instances. Further, out of 66 instances where Airshark failed to identify the device, 48 instances had RSSI values of less than −80 dBm and the rest involved frequency hopping devices with multiple other RF devices operating simultaneously. We observed a total of 12 (0.07%) false positives and these instances occured when operating multiple RF devices at low signal strengths.

Microbenchmarks

Airshark's detection accuracy is affected by low signal strengths and increased WiFi interference. Below we investigate these scenarios.

Performance under low signal strengths

We now highlight some of the reasons for reduced accuracy at low signal strengths by examining two of the features.

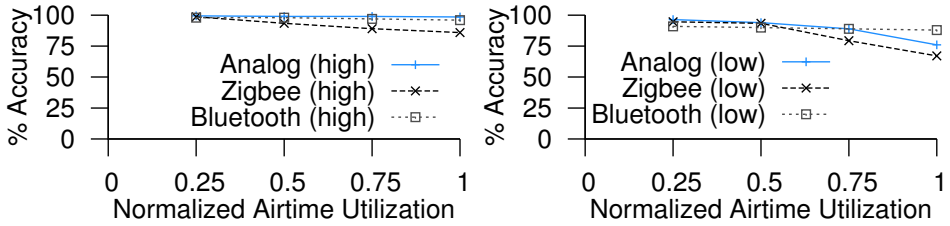

Figure 3.19: Stress testing Airshark with extreme WiFi interference. Detection accuracy is reduced for pulsed transmission devices (e.g., ZigBee), whereas accuracy for frequency hoppers is minimally affected.

— *Spectral signatures*. Consider a particular center frequency and associated bandwidth where we can expect an analog phone to operate. We wish to compute the spectral signature on this band (based on the received power in the FFT bins) and then measure the angular difference w.r.t. analog phone's spectral signature for (i) when the analog phone is active at this center frequency, (ii) when the phone is inactive. For Airshark to clearly distinguish between these two cases, there must be a clear separation between the angular differences *i.e.*, angular difference must be low when the phone is active, and higher when it is inactive. To understand the worst case performance, we also activate multiple other RF devices by placing them at random locations. Figure 3.18(a) shows that even in the presence of multiple devices, the angular difference is very low when the phone is operating at higher RSSI. However, when the phone is operating at lower signal strengths, the angular difference increases, thereby reducing Airshark's detection accuracy.

—*Bandwidth estimation.* In each run we activate a random RF device at a random location and let Airshark compute the bandwidth of the signal. Figure 3.18(b) shows the error in computed bandwidth at different RSSI values. We observe that Airshark performs very well at high RSSI values, but the bandwidth estimation error increases at low signal strengths, thereby affecting the detection accuracy.

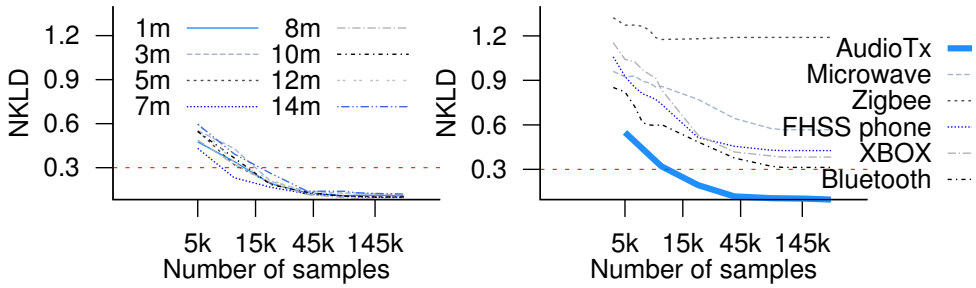

Figure 3.20: NKLD values wrt. to audio transmitter's pulse distribution for (a) audio transmitter at different distances (b) different RF devices

Performance under extreme interference

We performed stress tests on Airshark by introducing additional WiFi interference traffic. We placed a WiFi transmitter close to the Airshark node (distance of 1m) and let it broadcast packets on the same channel as the fixed frequency RF devices. We changed the WiFi traffic load resulting in different airtime utilizations. We tested the detection accuracy of RF devices at HIGH and LOW signal strengths (-50 dBm and -80 dBm respectively). Figure 3.19 shows the effect on Airshark's detection accuracy w.r.t. normalized air time utilization (air time utilization is maximum, when the transmitter broadcasts packets at full throughput). Accuracy of high duty devices (analog phone) is affected only in the LOW case, when normalized airtime utilization is close to 1. For devices like ZigBee (fixed frequency, pulsed transmissions), the effect is more severe in the LOW case under increased airtime utilization. Frequency hopping devices like Bluetooth, however, are not affected because Airshark is able to collect enough pulses from other sub-bands. It is worth pointing out that all the previous experiments were performed in the presence of regular WiFi traffic and in different wireless environments. We therefore believe that Airshark performs reasonably well under realistic WiFi workloads.

Choice of $_s$	-105 dBm	-95 dBm	-85 dBm
Accuracy (FPR)	97.3% (4.7%)	92.13% (0.041%)	89.24% (0.023%)

Table 3.7: Effect of different thresholds on Airshark's performance.

Deployment	**Proportion of RF device instances**				
	Microwave	Bluetooth	FHSS Phone	Videocam	Xbox
WLAN1	37.16%	52.34%	9.87%	–	0.6%
WLAN2	81.65%	17.43%	–	0.917%	–

Table 3.8: Proportion of non-WiFi RF device instances in 2 production WLANs. We collected data using Airshark for a duration of 48 hours.

Parameter tuning

We now discuss the empirically established parameters of our system. Table 3.7 shows the effect of using different energy thresholds. The set up for the experiments was similar to that in §3.4. We observe that while it is possible to improve Airshark's accuracy at lower RSSI values by lowering the threshold, this comes at the cost of increased false positives. Increasing the threshold reduces the number of peaks (and hence pulses) detected and reduces the detection accuracy.

We now show the effect of number of samples on the NKLD values. Figure 3.20 (left) shows how the NKLD values converge for an audio transmitter device (placed at different distances) with the total number of samples processed by Airshark. We find that around 15000 samples, the NKLD values converge to 0.3. Figure 3.20 (right) compares the NKLD of different RF devices when using the pulse distribution of the audio transmitter device as reference. We find that 15000 samples are sufficient, as NKLD values of $\leqslant 0.3$ can be used to indicate the pulse distribution of the audio transmitter.

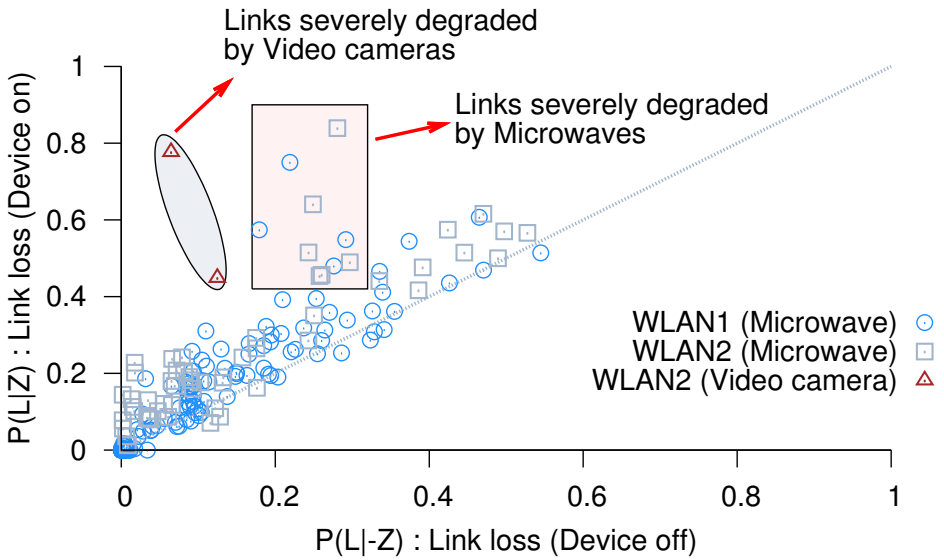

Figure 3.21: Results from a two day deployment of Airshark in two production WLANs. Each point in the scatter plot denotes the loss rate for a link in the absence ($p(L|\neg Z)$) and the loss rate in the presence ($p(L|Z)$) of (i) microwave ovens, (ii) video cameras, for a total of 224 links (168 links in WLAN1 and 56 links in WLAN2).

Example uses of Airshark

We now demonstrate Airshark's potential through example applications. We monitored the RF activity on a single floor of two production WLANs using regular wireless laptops that ran Airshark and also captured packets using *tcpdump*. WLAN1 used 7 APs and WLAN2 employed 3 APs on the monitored floor. We collected the data for 48 hours at each location. To provide confidence in our statistical estimates, we restrict our analysis to a total of 224 links (168 links in WLAN1, 56 links in WLAN2) that exchange at least 150 packets in our packet traces. Airshark can help WLAN administrators answer the following questions about RF device activity in their networks:

Airshark Trace		Loss rate	Airshark Trace		Loss rate
Duration	**Microwave**	**Link L1**	**Duration**	**Microwave**	**Link L2**
14:35:19—14:55:18	OFF	10.52%	16:50:36—16:53:19	OFF	16.66%
14:55:18—14:55:46	ON	86.20%	16:53:19—16:53:52	ON	79.61%
14:55:46—15:00:22	OFF	10.91%	16:53:52—17:13:22	OFF	15.78%
15:00:22—15:02:40	ON	45.88%	17:13:22—17:15:19	ON	48.78%
15:02:40—15:04:45	OFF	7.63%	17:15:19—17:33:20	OFF	15.00%

Table 3.9: Airshark traces for the time periods relevant to links L1 and L2 (WLAN1) showing increased losses due to microwave oven activity.

Question. "Which non-WiFi RF devices were visible in the WLANs? How long were the devices active, and which devices appeared more frequently?"

Analysis. Using traces from Airshark, we found that non-WiFi devices were active for 22.6% (10.48 hrs) and 13.92% (6.68 hrs) of the trace duration in WLAN1 and WLAN2 respectively. Table 3.8 shows the proportion of non-WiFi RF device instances in the two WLANs—microwave ovens (37.16% and 81.65%) and Bluetooth devices (52.34% and 17.43%) occurred most frequently in WLAN1 and WLAN2. FHSS cordless phones accounted for 9.87% of the instances in WLAN1. Game controllers and video cameras were also visible, albeit for very short durations.

Question. "Did any of the links in the WLAN suffer from interference due to non-WiFi devices? Which non-WiFi devices caused the most interference?"

Analysis. Airshark can help identify the interference-prone links as follows: Let L denote the event of a packet loss on a wireless link. We note that L might include losses due to non-WiFi interference as well as those due to "background losses" (e.g., due to weak signal)[4]. Let Z be the event that a non-WiFi device z is active. We compute the probability of a packet loss given the device is active,

[4]We note that intermittent losses may also occur due to potential hidden terminals in the network. We used PIE [149], a system that can detect hidden conflicts, to discard such cases from the traces.

$p(L|Z)$, and the probability of a packet loss given the device is inactive, $p(L|\neg Z)$ as follows:

1. Using Airshark, we identify the periods when the device z was active (t_{on}), and when the device z was inactive (t_{off}).
2. For each link, we compute the total number of packets transmitted on the link during t_{on}, and the corresponding number of packets lost, to measure $p(L|Z)$. Similarly, we compute $p(L|\neg Z)$ by measuring the loss rate during t_{off}.
3. For the links severely interfered by device z, we can expect $p(L|Z) \gg p(L|\neg Z)$.

We make the following observations:
— *Microwave Ovens:* Figure 3.21 shows the impact of microwave oven activity using a scatter plot of $p(L|Z)$ and $p(L|\neg Z)$. We observe increased losses for a few links (70—80%). We found that around 20% links in WLAN1 and 10% links in WLAN2 had more than 20% increase in loss rates. Further, for around 5% of the links, the loss rates increased by more than 40%. Table 3.9 shows snapshots of Airshark traces and loss rates for two links L1 and L2 that experienced interference from microwave ovens.
— *Video camera:* The camera was active only for around 3 minutes in the WLAN2 trace, but it had a severe impact on two of the links as shown in Figure 3.21. Losses for the two links increased from 6.47% to 77.67%, and 12.47% to 44.85% during the period the camera was on.
— *Bluetooth/Xbox/FHSS phones:* In both the traces, we did not find any impact of these devices on the link loss rates.

3.5 ISSUES AND DISCUSSION

Our work on Airshark is a first step towards detecting non-WiFi devices using commodity WiFi hardware, and is certainly amenable to various improvements. We now discuss some of the limitations and opportunities with detection approaches such as Airshark that we have not fully explored in this work.

- **Understanding the effects of sequential sampling on ground truth.** Due to driver limitations, current spectral sampling procedure implemented in Airshark is sequential in nature — Airshark scans the entire spectrum in a sequential order and spends an equal amount of time on each sub-band to collect samples. This implies that at any given instant, Airshark loses samples corresponding to all other sub-bands that are not being monitored. While we ran our training experiments in controlled settings with signal analyzers monitoring the spectrum, we currently are not able to capture the "entire device activity" (or the ground truth) for some of the devices. For example, for devices such as frequency hoppers (e.g., Bluetooth or XBox controllers) we are only able to capture partial activity corresponding to one sub-band at a time. Losing samples affects the device detection time as it increases the time to capture features such as the pulse spread (Section 3.3). One way forward is to use multiple Airshark nodes each statically monitoring a particular sub-band and merging the samples to construct the entire device activity. We note that such an approach would require the clocks on Airshark nodes to be time synchronized (See Chapter 7). Capturing the entire device activity would then help us benchmark the sequential sampling procedure and understand the scope for improvement in Airshark's accuracy. This will also be useful in benchmarking other sampling approaches described next.

- **Improving sampling procedure.** We note that sequential sampling approaches are sub-optimal as spectrum might not be equally utilized. For example, some parts of the spectrum might be more crowded (due to high activity from a particular non-WiFi device, or due to the presence of multiple non-WiFi devices in this portion of the spectrum). A better sampling approach would be "sense" the parts of spectrum that are more crowded, and *adaptively* tune the dwell times on the sub-bands with higher non-WiFi device activity. Examples of such approaches include active sensing mechanisms [35] that are adaptive in nature. Newer sampling mechanisms such as distilled sensing [63] approaches are also of interest.

These approaches are based on the notion that it is often easier to rule out parts of the spectrum that do not contain signal than it is to directly identify non-zero signal components. We expect such adaptive sampling mechanisms to improve Airshark's detection accuracy and the time to detect each of the active devices.

- **Handling multiple overlapping signals.** Currently, Airshark can only handle a limited number (up to 6) of simultaneously active devices. A limiting factor in Airshark's detection capabilities in presence of multiple, simultaneously operating non-WiFi devices is that of "overlapping signal scenario" — a scenario where signals from multiple devices that always happen to overlap in both time and frequency domain. We find that our extensions to handle such cases (e.g., using partial signatures) are not particularly effective, resulting in degradation in detection accuracy. Instead of using partial signatures, another approach would be to treat the received samples as a addition of individual known signals (since total power of the received signal is equal to the sum of the powers of the constituent signals and noise) and train the classifier for possible overlap combinations of different devices [66]. There is also a scope for applying signal de-noising approaches [47, 167] that can prove to be beneficial in such overlapping signal cases.

3.6 SUMMARY OF AIRSHARK

In this chapter, we first motivated the need to detect non-WiFi RF devices by characterizing their prevalence in typical environments. We then presented Airshark, a system that can detect the non-WiFi devices using only the functionality provided by commodity WiFi cards. Airshark extracts unique features using energy samples from a WiFi card and presents a generic, and extensible framework to accurately detect multiple non-WiFi devices, while maintaining a low false positive rate. We also found its performance to be comparable to a commercial signal analyzer. Through a deployment in two

production WLANs, we demonstrated Airshark's potential in understanding non-WiFi interference issues.

4 DECONSTRUCTING NON-WIFI INTERFERENCE USING WIFI HARDWARE

4.1 MOTIVATION

In the previous chapter, we showed how a WiFi device can use information from emerging wireless cards to *detect* the presence of non-WiFi devices operating in its vicinity. In this chapter, we design *WiFiNet* — a collaborative neighborhood of WiFi nodes — to identify and "catch" the non-WiFi transmitters that are actually causing harmful interference to WiFi communication (Figure 4.1). More specifically, through WiFiNet we can answer the following questions — *how much* interference is any non-WiFi RF transmitter (e.g., a Bluetooth headset, an active analog phone, or a microwave oven) causing to an existing WiFi communication and *where* in the physical space is each such non-WiFi interferer located?

Much of the prior work has employed custom hardware to tackle non-WiFi interference. Examples include commercial products such as AirMaestro [3] and Wispy [12] that build specific signatures to *detect* the presence of a device. Recent research efforts (e.g., RFDump [98], DOF [66], TIMO [139]) have used the flexibility allowed by software radios to develop novel signal processing techniques and physical layer designs to co-exist with these devices. The unique aspect of WiFiNet is that it is built entirely on top of standard WiFi network interface cards (NICs). In particular, an emerging class of WiFi NICs, such as those based on the Atheros 9280 chipset, as part of their WiFi frame decoding process, provide coarse-grained energy samples per sub-carrier of a WiFi channel. These energy samples are a few orders of magnitude lower in resolution than available to the sophisticated spectrum analysis tools. Using such functionality, in the previous chapter we presented Airshark [131] and have shown that even with such a low resolution input, a regular WiFi node (either an Access Point or a client) can individually *detect* the presence of non-WiFi devices.

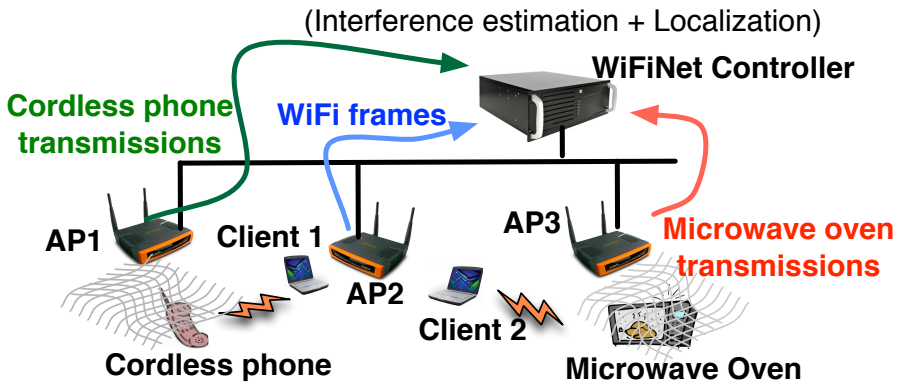

Figure 4.1: Illustration of WiFiNet's architecture.

Airshark is, however, is only the first step in the broad space of deconstructing non-WiFi interference and quantifying their impact on WiFi links. WiFiNet leverages collaboration between multiple WiFi nodes to address both quantification of interference impact and localization of these interferers, as we explain below.

Quantifying non-WiFi interference impact in real-time: The mere presence of a non-WiFi device, as detected by Airshark, in the vicinity of a WiFi transmitter is not always harmful. For instance, an active analog cordless phone at a specific location, may only have a minimal impact on a particular WiFi link. We call such a low-impact non-WiFi device, a *minnow*. On the other hand, a microwave oven radiating a significant amount of energy in its vicinity might cause severe disruption to nearby WiFi links. We call such an interferer, a *whale*.

However, the impact of interference from the same non-WiFi device can quickly change over time. For instance, if the microwave oven's setting is adjusted to operate with a low power level, this device may suddenly turn into a minnow. On the other hand, if the cordless phone user moves to a different location which is closer to the WiFi link, this device might turn into a whale with respect to this WiFi link. It is even possible that the impact of the cordless phone on the WiFi link changes due to properties of the WiFi link itself. For example,

when the WiFi link is operating at 54 Mbps, the disruptive impact of the cordless phone is quite high, with the impact decreasing as a rate adaptation algorithm reduces the WiFi link's choice of PHY rates. WiFiNet tracks this continuously changing impact of non-WiFi transmitters on WiFi communication in real-time, adjusting its interference estimates immediately as operating parameters change (e.g., the microwave power setting is changed, or the WiFi device's PHY rate selection algorithm starts operating with a higher rate).

Locating non-WiFi interferers: WiFiNet also determines the physical location of such non-WiFi transmitters immediately, so that the precise source of such interference can be determined, and if needed, such interfering devices can either be re-configured or disabled.

Through these new and unique capabilities, WiFiNet provides new RF management tools for WiFi environments using off-the-shelf WiFi NICs only, obviating the need for sophisticated wireless hardware. In fact, WiFiNet can be easily implemented and integrated into enterprise WiFi APs to achieve improved mitigation strategies against non-WiFi interference for enterprise environments.

Challenges in designing WiFiNet

In designing and implementing the capabilities of WiFiNet, we had to overcome the following set of challenges:

How to detect multiple devices of the same type? In many wireless environments, there are multiple devices of a given type, e.g., two different cordless phones. It is possible that among these two phones, one is a whale and causes 80% loss in throughput to a WiFi link, while the other is a minnow and causes only 5% loss in throughput. To differentiate between these two interferers, WiFiNet needs to determine how many devices of each type are operating at any given instant. To achieve this goal, WiFiNet utilizes tight clock synchronization, and employs signal clustering techniques operating on some device specific attributes (when available) and signal strength observations gathered by multiple WiFi detectors to identify the unique transmission

contributions from different, potentially identical, non-WiFi devices. Our prior work, Airshark, builds signatures of each device type to detect the presence of any such device in the vicinity of the detecting WiFi node. But such an individual WiFi node is not able to determine if there is only one or two or three different FHSS cordless phones in the vicinity, and hence, cannot attribute which part of wireless transmissions belong to which such interferer.

How to estimate each device's impact? After segregating each non-WiFi device's transmissions, WiFiNet uses specific timing analysis for estimating the impact of each interferer — time-frequency overlaps between the WiFi frames and non-WiFi device's transmissions are analyzed and correlated with the outcomes (frame success or loss) to discern the impact of each device. Our technique works well for both low and high duty devices. In our design, we take into account the carrier sensing interference, interference from WiFi sources and multiple PHY rates of operation used by WiFi links.

How do we localize the non-WiFi device? Localization in indoor wireless environments is a well studied problem [25, 29, 140, 166]. Common techniques include signal strength based triangulation [166] and RF fingerprinting approaches [25]. However, the key requirement for such localization approaches is for multiple detectors to *detect the same transmission* at different signal strengths. In the commonly known WiFi localization techniques, this is easy because the different detectors decode the same wireless frame and use the frame's identity to ensure sameness.

In our case, the WiFi detectors cannot decode the non-WiFi transmissions, and hence cannot immediately assign the same identity to "pulses" received from the non-WiFi transmitters. A core challenge that we needed to solve is for different WiFi detectors to determine which received pulses correspond to a single transmission from the same non-WiFi device. The next challenge is to build a model for localization. Propagation characteristics are similar for both WiFi and non-WiFi transmitters since they operate on the same frequency. WiFiNet exploits this fact and builds the model by exchanging WiFi frames and recording signal strength measurements. Since the transmit power of non-WiFi devices can be arbitrarily different from that of WiFi nodes, the model takes this into account by operating on the *difference* in received signal strengths.

Through experiments, we show the feasibility of this approach for non-WiFi device localization using WiFi-only detectors.

Summary of key contributions: Summarizing, the key contributions of our WiFiNet system are three-fold: (i) it detects and discerns the transmission contributions of different non-WiFi interferers in the vicinity of the WiFi detectors; (ii) it attributes interference impact of each such non-WiFi device for any given WiFi link, classifying them as whales, minnows, or anything else in between, through collaborative observations; and (iii) it pinpoints the location of each such non-WiFi interferer so that they can be independently re-configured or disabled. All of these capabilities are implemented using WiFi-only detectors.

The entire WiFiNet system has been implemented using the Atheros AR 9280 based WiFi NICs, and evaluated in detail through various experiments. Our results indicate a typical impact determination accuracy of $> 90\%$ and a localization error of < 4 meters in these environments.

4.2 WIFINET

We start by presenting an overview of WiFiNet's architecture, followed by the details of its design and operation.

Architecture and flow of operations. WiFiNet employs collaborative observations from multiple WiFi-only detectors spread across a network to perform its non-WiFi device interference estimation and localization operations. Since most enterprise APs today come equipped with multiple WiFi radios, one way to deploy WiFiNet would be to employ one of the radios as a detector. In such a setting, WiFiNet can function as follows. All the enterprise APs are connected to a central controller over an Ethernet backplane. Each AP has two radios: (i) a regular radio that used to communicate with the clients, and (ii) a detector radio that continuously captures spectral samples as well as WiFi frames. APs run Airshark [131] to process the spectral samples and perform device detection. Airshark outputs a set of "pulses" (time-frequency blocks representing non-WiFi device transmissions), and tags these pulses with the appropriate device type (e.g., Bluetooth, ZigBee, etc.). Each pulse reported

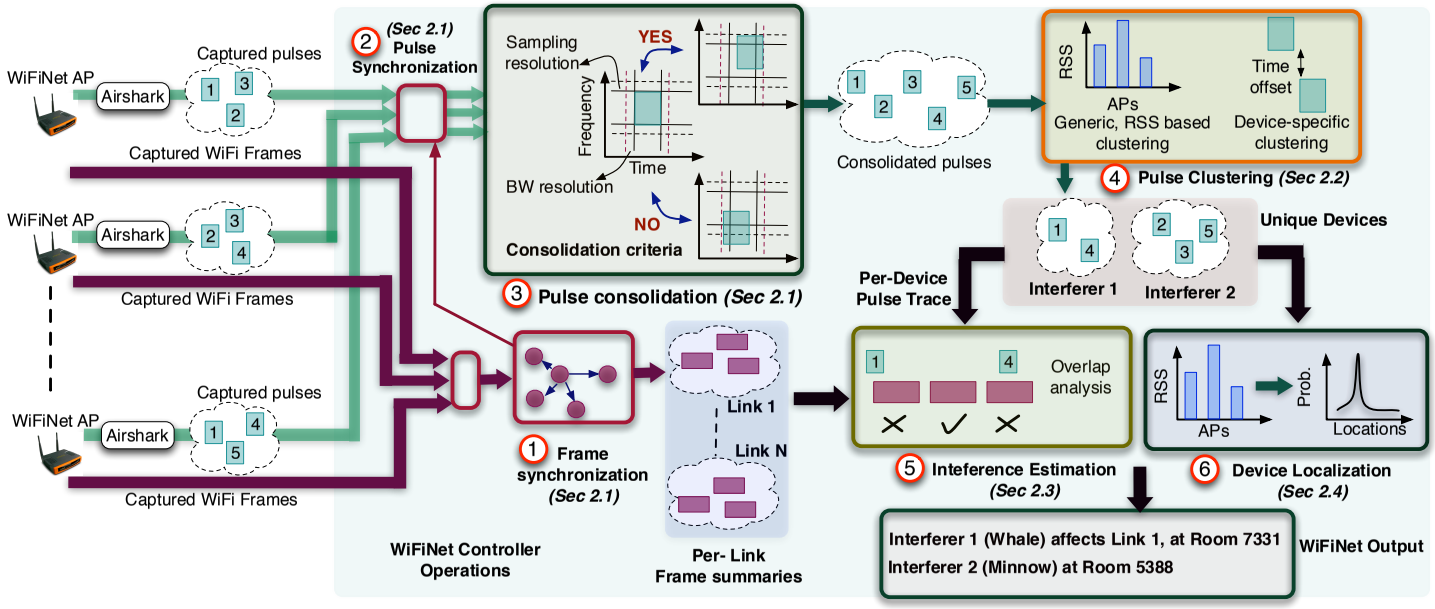

Figure 4.2: Flow of operations in WiFiNet. WiFiNet APs capture spectral samples as well as WiFi frames. Each AP runs Airshark [131] to detect non-WiFi devices and output non-WiFi pulses (transmissions) tagged with device type. WiFi frames are used to synchronize the clocks at the APs. Synchronized clocks at the APs are then used to consolidate the pulses across multiple APs using a heuristic (§4.2). Consolidated pulses are then clustered using (i) RSS based clustering and (ii) device-specific clustering methods to output unique non-WiFi device instances and their pulses (§4.2). For each non-WiFi device instance and WiFi link, the interference detection module then analyzes the impact of the device on the link using transmission overlaps (§4.2). Model-based localization algorithms are used to localize each non-WiFi device instance (§4.2).

by Airshark consists of the start and end timestamps, center frequency and bandwidth of the pulse, the average received power of the pulse, and a tag that indicates the device type. Next, the APs also process the captured WiFi frames to create a per-client frame transmission summary: frame start and end timestamps, PHY rate, and reception status (*i.e.*, whether the AP received an ACK for this frame or not). The proximity between the two radios ensures that the detector radio receives the majority of frames transmitted by the regular radio due to capture effect, thereby creating an accurate summary of frame transmissions [149]. The per-client WiFi frame transmission summaries and the captured non-WiFi pulse traces are forwarded to controller to identify the individual non-WiFi device instances, estimate their interference impact and localize them. Figure 4.2 presents the overall control flow. We now explain each of these tasks in detail.

Identifying unique pulses

Since the same pulse can be received by multiple APs in the WLAN, the first task for the controller is to consolidate the traces and identify the *unique pulses* transmitted by different non-WiFi devices operating in the environment. To do this, the controller has to identify the "common" pulses received by the APs and create a single consolidated pulse. However, finding common pulses is not straightforward as WiFi APs *cannot decode* non-WiFi pulses.

Pulse consolidation. WiFiNet uses a heuristic to consolidate the pulses: if two APs receive a pulse that has the same device type (e.g., Bluetooth), has the same start and end times, has the same center frequency and bandwidth, then most likely the APs received the same pulse (transmitted by a particular non-WiFi device). In practice, we allow a certain leeway as these parameters might not exactly match e.g., we allow the maximum difference between the pulse start (and end) times to be FFT sampling resolution of the WiFi card (116μs for AR9280 card) and that between pulse center frequencies (and bandwidths) to be resolution bandwidth of the WiFi card (312.5 kHz or equal to 802.11 sub-carrier spacing).

To apply the heuristic, however, would require the pulse traces at the APs to be synchronized. How do we synchronize the pulse traces without knowing common pulses (*i.e.*, reference points)? WiFiNet solves this issue by leveraging the WiFi hardware — the timestamps of the pulses are derived from the *same clock* that is used to timestamp the captured WiFi frames. WiFiNet first synchronizes the clocks at all the APs using captured "common" frames as reference points, and then uses the synchronized APs to find "common" pulses. We now explain these tasks.

Opportunistic synchronization. Synchronization can be easily carried out if we find one reference frame that is received by all WiFiNet APs in the WLAN. However, this is highly unlikely in practice as the wireless signal attenuates over distance. Therefore, WiFiNet *opportunistically synchronizes* 'pairs of APs' using common received frames (reference frames used for synchronization are typically beacon frames or data frames without the retransmit bit set [38, 123]) , and then *transitively* synchronizes the sniffer radios at all the APs using a graph based approach:

1. Instantiate a synchronization graph, $syncGraph = (S, E)$ where each vertex s_i represents the sniffer radio at WiFiNet AP i, and the weight $w(e_{ij})$ of the edge $e_{ij} : s_i \rightarrow s_j$ represents the clock skew between the APs. Initially, S comprises of all vertices and has no edges.
2. For each pair of WiFiNet APs (s_i, s_j), we start by finding the set of reference frames (common received frames). Compute difference in timestamps for each of the reference frames, and use the median value as the clock skew. This results in instantiating an edge e_{ij} in the *syncGraph* with weight equal to skew between the APs.
3. After processing all pairs of APs, start with a reference AP (s_0), chosen randomly, and perform a *breadth-first search* on the *syncGraph* to transitively synchronize all the APs with respect to the reference AP.

The above synchronization procedure is repeated every *sync interval* in order to account for the clock drift. Figure 4.3 shows the synchronization error as a

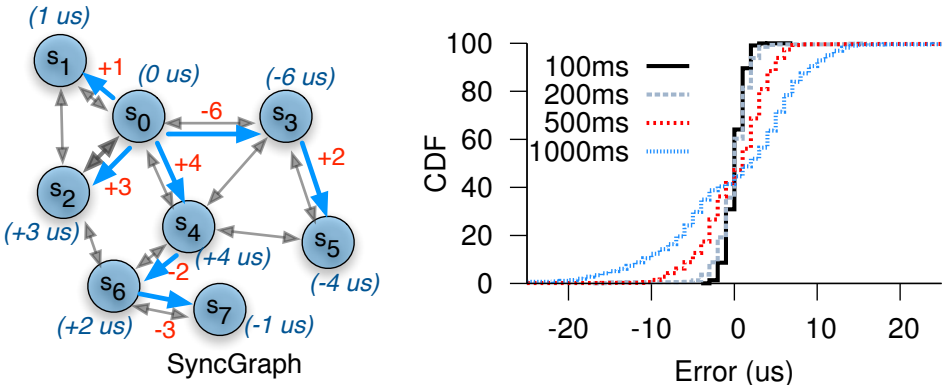

Figure 4.3: (left) Illustration of graph based opportunistic synchronization used in WiFiNet. Each node is an WiFiNet AP (s_0 is the reference AP), weights on the edges correspond to the pair-wise AP skews, and the numbers in the parentheses are the final synchronization offsets of the APs (*i.e.*, skews w.r.t. reference s_0) (right) CDF of synchronization error for our deployment of 8 WiFiNet APs at different sync intervals. Error in synchronization is $\leqslant 6$ µs for a sync interval of 500 ms.

function of sync interval for our deployment of 8 WiFiNet APs. We observe that the error increases with the increase in sync interval (e.g., an error of < 6 µs for an interval of 500 ms). In WiFiNet, we use a sync interval of 100 ms which results in tight synchronization between the APs (an error of less than 2—4 µs in most cases).

Output from consolidation. The controller applies the appropriate synchronization offsets to each AP's pulse trace and then finds the common pulses among the APs using the heuristic mentioned above. The consolidation process can be carried out efficiently as the pulses are sorted by time. After consolidation, the controller is left with unique pulses transmitted by non-WiFi devices, and for each unique pulse, we associate an RSS vector $\mathbf{r} = [r_0, \ldots, r_{N-1}]$ that represents the received power of this pulse at each of the N APs in the WLAN. We set r_i to the average received power of the pulse at ith AP, if the pulse was indeed received this AP, otherwise $r_i = \phi$.

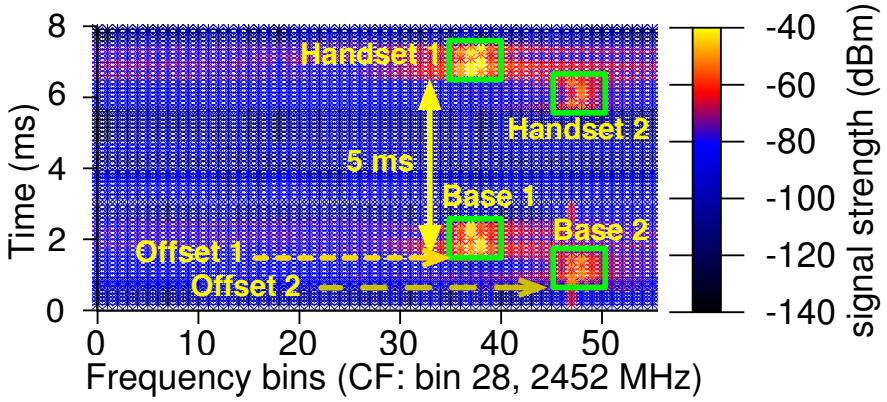

Figure 4.4: Heatmap of 4 FHSS cordless phone devices (2 base/handset pairs) captured by a WiFiNet AP, showing the timing property. Each base/handset pair emits two short pulses that are both at the same center frequency and are separated by 5 ms. The pair then jumps to a different center frequency after 10 ms and repeats the process. WiFiNet identifies the pulses belonging to each device by calculating their timing offsets (pulse start time modulo 10 ms).

Identifying unique device instances

After obtaining the unique pulses, the next task for the controller is to detect the number of non-WiFi device instances, segregate the pulses belonging to each instance and establish a unique ID for it. WiFiNet first segregates the pulses according to their device type, and employs *clustering algorithms* for further segregation. The algorithms determine "the number of clusters" (non-WiFi device instances), and assign each pulse to a cluster. The combination of (device type, cluster center) is then used as the ID for this device instance. In our current prototype, we implement (i) a generic, RSS based clustering that is applicable to all non-WiFi devices and (ii) clustering based on timing properties that is specific to some non-WiFi device types. We now explain both approaches.

Generic clustering based on signal strength

WiFiNet's generic clustering approach operates on RSS vectors that are N-dimensional (*i.e.*, vector sizes grow with the number of APs). Since the

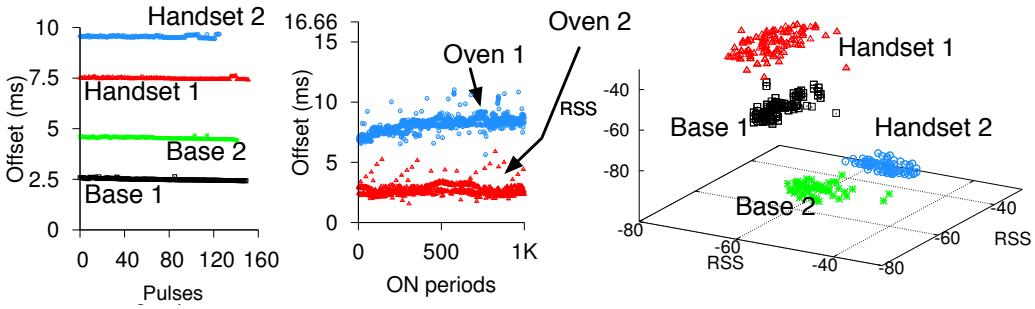

Figure 4.5: Segregating pulses in the presence of multiple, simultaneously operating devices of the *same type*, based on WiFiNet's device specific and generic clustering. Figure shows clusters of pulses from (left) 2 FHSS cordless phone base/handset pairs (4 FHSS cordless devices) using pulse start time offset (middle) 2 Microwave ovens using ON-period offset (right) 4 FHSS cordless devices using a generic, RSS based k-means + EM-clustering technique using 3 WiFiNet APs.

performance of clustering mechanisms typically degrades with increase in the number of dimensions, we first filter out some of the dimensions before performing clustering.

Reducing vector dimensions. We use some optimizations to reduce the number of dimensions: (i) clustering is performed every *scan window* (5 secs in our current prototype) to keep the number of pulses low (ii) APs not receiving any pulse in the current window are discarded (iii) The controller uses the *syncGraph* (§4.2) as a proxy for "RF neighborhood" of APs and segregates the APs into different partitions using the following heuristic. For each pulse, we determine the WiFiNet AP with the highest RSS, and increment a counter for this AP. After processing all the pulses, we assign the AP with the highest counter to first partition. We then pick each AP (in the decreasing order of the counters) and place it in one of the existing partitions if it satisfies the RF neighborhood constraint: in the *syncGraph*, the AP has to be within a 2-hop neighborhood of the AP with the highest counter for this partition, otherwise we create a new partition for this AP. This breaks the clustering problem into

sub-problems, each operating on one partition.

Handling missing values. After partitioning the RSS vectors to reduce the dimensionality, another problem remains: RSS vectors might still have missing values (*i.e.*, $r_i = \phi$) for some columns. This is because APs might capture pulses intermittently (i) as they are far from the device, or (ii) due to a stronger signal from other WiFi or non-WiFi transmissions [131] that overlapped with the pulse. While it is possible to define a distance function for clustering that ignores missing values in the vectors, such a function is unsuitable for many traditional clustering algorithms as it doesn't satisfy certain mathematical properties such as the triangle inequality [70]. This presents us with two choices, (i) use clustering algorithms which allow a certain degree of freedom in the formulation of a suitable distance function or (ii) fill in the missing values using a best-effort approach, and then use traditional clustering algorithms. We explored both these choices.

(Method 1) Density-based clustering. We used DBSCAN [72], a density-based clustering approach that allowed us to formulate a distance metric that can handle missing values. Let P and Q be the set of APs receiving the pulses p and q, and C be the set of common APs that received both pulses. We define $\eta = |C|/\max(|P|, |Q|)$ and compute the distance between two pulses p and q as:

$$\tau(p,q) = \begin{cases} \sqrt{\dfrac{1}{|C|}\sum_{i \in C} [r_i^{(p)} - r_i^{(q)}]^2} & \text{if } \eta \geqslant \eta_o, |C| \geqslant C_{min} \\ +\infty & \text{otherwise} \end{cases}$$

The above measure only takes signal strength from common APs into account, and any missing RSS values do not affect the distance. Intuitively, the introduction of parameters η_o and C_{min} is to account for the case when the difference in the set of APs receiving the pulses is too large. We comment on these parameters in §5.7.

(Method 2) k-Means + EM-clustering. Another approach to handle missing values is to first perform *imputation* — missing values in a particular column are replaced (e.g., using a median or mode of the column). In WiFiNet, we use *EM-Imputation* [11], a well known imputation method, where the missing values are replaced by using expectation maximization with a multi-variate normal model. After imputation, we can use traditional clustering mechanisms as the distance function (e.g., Euclidean) can now operate on all the columns of the vectors. We experimented with several clustering algorithms and found that a combination of k-Means and EM-clustering perform the best: WiFiNet controller iteratively runs the k-Means clustering algorithm with different values of k ($1 \leqslant k \leqslant k_{max}$), and then picks the best solution [11]. This is used as the initial solution to the EM-clustering algorithm, which outputs the final non-WiFi device instances and the corresponding pulses. In our experiments, we set $k_{max} = 10$ *i.e.*, we assume that the maximum number of *simultaneously operating* devices of the *same type* to be 10. Figure 4.5 (right) shows an example result for RSS based clustering for 4 FHSS cordless phone device instance using 3 WiFiNet APs. In §5.7, we compare the performance of the above clustering algorithms.

Clustering based on device specific attributes

We found that some non-WiFi device types exhibit certain specific timing properties that can be exploited to provide better clustering performance compared to the generic RSS based clustering approach. In WiFiNet, we implemented such clustering for two non-WiFi device types:

— *Pulse start time offset for FHSS cordless phones.* WDCT cordless phone sets cycle through frames of 10 ms: each frame consists of two short pulses, one emitted by the base at the beginning of the frame and the other by the handset, occurring after 5 ms (both at the same center frequency). Both base and handset then jump to a different center frequency for the next frame. Figure 4.4 shows the pulses from two cordless phone sets (*i.e.*, 2 base/handset pairs, a total of 4 unique cordless phone devices) captured by WiFiNet. Figure 4.5 (left) shows that clustering based on the pulse start time offsets ($t \bmod 10$) can segregate the pulses belonging to each device.

— *ON-period offset for microwave ovens.* Microwave ovens emissions exhibit an ON-OFF pattern, typically periodic with a frequency of 60 Hz (frequency of the AC supply line) *i.e.*, a period of 16.66 ms [131]. WiFiNet computes the offset for start times of the microwave pulses (ON periods) as $t \bmod 16.66$ and uses this to segregate their pulses. Figure 4.5 (middle) shows the result of clustering pulses from two microwave ovens operating simultaneously.

Interference Estimation

After clustering, WiFiNet controller now has a set of clusters, each representing a unique non-WiFi device instance. We now explain how the controller can analyze the interference impact of each device instance.

Intuition and Overview. For each non-WiFi interferer instance and a WiFi link, WiFiNet controller performs interference analysis by correlating the the link's frame transmission with the non-WiFi device's pulse transmissions and observing the reception status of the frames. It measures the impact of a non-WiFi device on a WiFi link by computing the probability of a frame loss when the frame overlaps with a simultaneous transmission from the non-WiFi device. Intuitively, the extent of interference is directly proportional to the probability of losing overlapping frames. This allows WiFiNet to maintain a *continuous interference model*, where the extent of interference can be any value between 0 and 1. For instance, in Figure 4.6, the controller observes that frames transmitted on a link are *unsuccessful* whenever a non-WiFi device's *pulse overlaps with the frame in time (and frequency) i.e.*, frames F_1 and F_3 overlap with non-WiFi device pulses T_1 and T_2, and are lost. It can therefore infer that the device strongly interferes with the link. Such fine-grained timing analysis is possible because APs are tightly synchronized (§4.2) and they use the *same clock* to timestamp both pulses and the frames. We now explain our interference estimation metrics.

Metrics for interference estimation. Formally, the interference estimation metrics used in WiFiNet can be explained as follows. Let I be the event that interference from a particular non-WiFi device causes a frame transmission

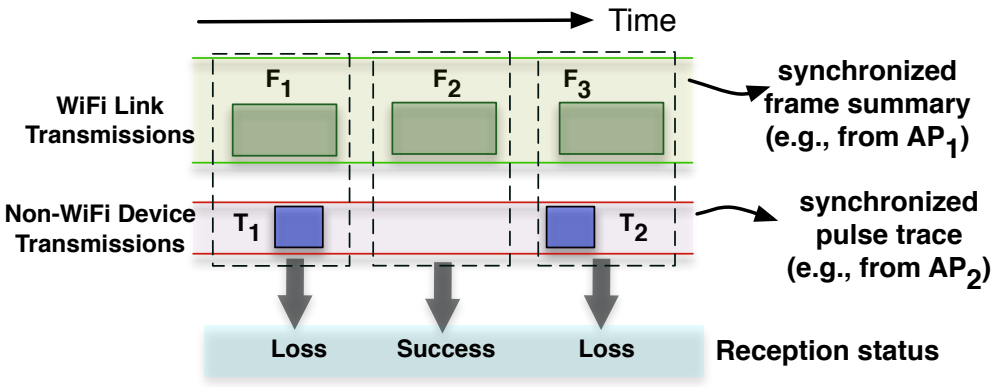

Figure 4.6: Illustration of interference estimation in WiFiNet.

to be unsuccessful. Let L be the event of an unsuccessful transmission due to background losses (e.g., due to weak signal) and O denote the event of an overlap between the frame transmission and a simultaneous transmission (e.g., a pulse) from the non-WiFi interferer.

— *(Metric 1) Impact given overlap.* Conditional probability, p[I|O] is used to measure the *impact given overlap i.e.*, probability that a frame is unsuccessful given an overlap with a simultaneous transmission from a non-WiFi device.

— *(Metric 2) Overall impact*. WiFiNet also maintains p[I], the *overall impact* of a non-WiFi device. Here, p[I] is equal to p[I|O]·p[O] (when there is no overlap, p[I|¬O] is simply 0). That is, p[I] is probability of frame loss due to the overall activity from the non-WiFi device.

We note that p[I], the overall impact of the interferer, depends on the probability of overlap p[O], which varies based on the link and interferer transmission patterns. Whereas, p[I|O] is *not affected* by these transmission patterns *i.e.*, p[I|O] indicates the *worst case impact* of the interferer on the link, which is observed when p[O]=1 (*i.e.*, when the transmissions of link and the interferer always happen to overlap). Next, we explain how these probabilities are estimated by WiFiNet in real-time.

Interference estimation. The controller measures the total number of frames transmitted (n) on the WiFi link of interest, the number of frames that overlapped with the non-WiFi device's transmissions (n_o) and n_o^l, the number of overlapped frames that were unsuccessful. It then computes $p[O]$, the probability of transmission overlap as n_o/n. Next, the controller computes $p[(I \cup L)|O] = n_o^l/n_o$ *i.e.*, the probability of an unsuccessful frame transmission due to either background losses or interference from the non-WiFi device, given an overlap in transmissions. It also computes the probability of frame loss when there is no overlap from the interferer, $p[L]$ as n_{no}^l/n_{no}. Here, $n_{no} = n - n_o$ is the number of frames without overlap and n_{no}^l is the number of n_{no} transmissions lost. Since L is independent of O, we have $p[L|O] = p[L|\neg O] = p[L]$. Also, I and L are independent events, and so we have $p[(I \cup L)|O] = p[I|O] + p[L] - p[I|O] \cdot p[L]$. That is,

$$p^{\text{WiFiNet}} = p[I|O] = \frac{\Big(p[(I \cup L)|O] - p[L]\Big)}{(1 - p[L])} \tag{4.1}$$

Using $p[(I \cup L)|O]$ and $p(L)$, WiFiNet controller estimates $p[I|O]$. Following this, the controller also computes the overall interference $p[I]$ as $p[I|O] \cdot p[O]$.

Handling overlaps from multiple non-WiFi interferers. In general, a frame transmission may overlap with multiple simultaneous transmissions from potential non-WiFi interferers. In this case, WiFiNet controller attributes the frame transmission success or loss to each overlapping non-WiFi interferer. We observed that *diversity* in the frame transmission times [149] as well as the diversity in transmission times of different non-WiFi devices allows WiFiNet to distinguish the *true* non-WiFi interferer from the other *false* non-WiFi interferers (*i.e.*, devices that happened to transmit at the same time as the true interferers). In particular, such a diversity allows WiFiNet to observe further transmissions from false non-WiFi interferers that overlap with the frames but do not lead to a frame loss. In our experience, such a transmission diversity arises due to (i) distinct transmission characteristics of different non-WiFi devices (e.g., frequency hopping devices typically emit short pulses at different center

frequencies) and (ii) diversity in the usage times of non-WiFi devices [131], where in a typical enterprise not more than 3−4 devices were found to be *simultaneously active*.

Enhancements to the basic technique

Handling high duty devices operating with other devices. Transmissions from multiple devices that *always* happen to overlap in time can lead to cases where WiFiNet can make incorrect estimates. For example, WiFiNet may identify a false interferer as a true interferer if the transmissions from the false interferer always happen to overlap with that of a true interferer. In our experiments, we found that such a scenario is unlikely when using pulsed transmitters (e.g., ZigBee devices) or frequency hopping devices (e.g., Bluetooth or FHSS cordless phones) that typically emit short pulses. However, operating high duty devices (e.g., analog cordless phones) that *continuously* emit energy alongside other non-WiFi devices will cause their transmissions to always overlap that can lead to incorrect estimates (§4.3).

We use two refinements to the basic approach to correctly identify interference impact of a non-WiFi interferer W operating alongside a high duty device H: (i) when computing $p[I_H|O_H]$ for a high duty device, we only consider the frames that *do not overlap* with a transmission from any other non-WiFi device. Here, $p[O_H] = 1$ and $p[I_H|O_H] = p[I_H]$ and (ii) we modify Equation 4.1 to compute the estimate $p[I_W|O_W]$ of a non-WiFi device W when it operates alongside a high duty device H as

$$p[I_W|O_W] = \frac{\Big(p[(I_H \cup I_W \cup L)|O_W] - p[I_H \cup L]\Big)}{\Big(1 - p[I_H \cup L]\Big)} \tag{4.2}$$

Here, $p[(I_H \cup I_W \cup L)|O_W]$ can be computed by measuring the losses that happen when the frames overlap with transmissions from non-WiFi device W (as well as those from the high duty device H). Now, knowing $p(I_H)$ and $p(L)$, we can compute $p[I_W|O_W]$. While such an approach can handle most cases, it cannot handle pathological cases where two high duty devices are activated at the exact same time — since both the devices continuously emit

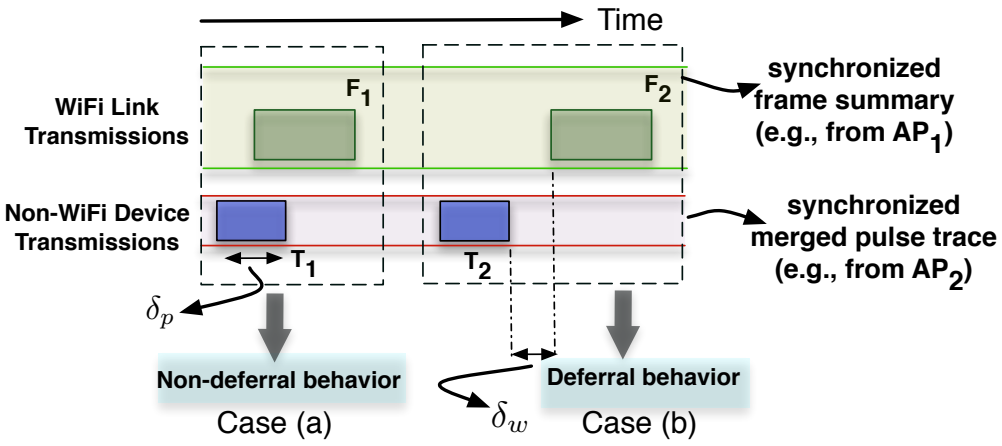

Figure 4.7: Illustration of carrier sensing estimation in WiFiNet.

energy, nothing useful can be said about the impact of each device (§4.3). If however, the interference impact of one of the devices is known, then that of the other can be computed using the above formulation. As we show later in §5.7, typically, diversity in transmission times of non-WiFi devices, coupled with the above refinements allows WiFiNet to correctly identify the true non-WiFi interferer in realistic wireless settings.

Quantifying impact at different 802.11 rates. The impact of a non-WiFi interferer on a WiFi link also depends on the PHY rate being used by the WiFi transmitter. To account for this, WiFiNet controller records the overlaps and losses separately for each different PHY rate, and computes a separate interference estimate for each rate. This helps quickly the estimate interference impact at each PHY rate when using dynamic bit rate adaptation as opposed to high-overhead bandwidth tests that require controlled experiments at each PHY rate to estimate the same [149].

Handling sender-side interference. Similar to the procedure used for interference analysis, WiFiNet controller can infer whether a WiFi transmitter is deferring to a non-WiFi device by correlating the WiFi frame transmissions

with the non-WiFi pulse transmissions. Figure 4.7 shows two cases of interest. Case(a) when the WiFi transmitter is not deferring to the non-WiFi device, WiFiNet controller will observe several instances where the frame transmission *starts while the pulse transmission is in progress*. Case(b) When the WiFi transmitter is indeed deferring to the non-WiFi device, WiFiNet controller will not observe instances where frame transmission starts while the pulse transmission is in progress. However, this condition alone is not enough to infer that the WiFi transmitter is deferring to the non-WiFi device, as it may happen that the WiFi transmitter did not have any packets to send while the pulse transmission was in progress *i.e.*, the WiFi transmitter did not contend for the medium. To identify the deferral instances, we use a heuristic similar to the prior work on carrier sense estimation between WiFi links [123, 149]: the controller identifies the deferring frames as those where the difference between the pulse transmission end time and the frame transmission start time is within a certain threshold δ_w. Here, δ_w is the maximum time spent by the WiFi transmitter performing back-off and is set to $28 + 320$ µs (DIFS + Max back-off period for 802.11g).

The controller can now compute the fraction $\Delta_{cs} = \frac{n_d}{n_d + n_{nd}}$ where n_{nd} is the number of Case (a) instances that indicate *non-deferral* behavior and n_d is the number of Case (b) instances that indicate *deferral* behavior. If the transmitter is indeed deferring, Δ_{cs} would be close to 1. Whereas, if the transmitter is not deferring to the non-WiFi device, the difference in the pulse and frame start transmission times would be uniformly distributed in the interval $[0, \delta_p + \delta_w]$, where δ_p is the duration of the pulse. That is, we expect $\Delta_{cs} \approx \frac{\delta_w}{\delta_p + \delta_w}$. Typically, $\delta_p > \delta_w$, therefore Δ_{cs} is low for cases of non-deferral (e.g., for δ_p for microwave ovens, cordless phones, and Bluetooth devices is 8 ms, 1.25 ms, and 625 µs respectively). In our experiments, using a threshold of $\Delta_{cs} > 0.8$ was able to correctly identify deferring WiFi transmitters (§4.3).

Extensions to handle WiFi interference. In general, WiFi links can also experience interference from other WiFi links. We extend our basic approach to measure the overlaps between frame transmissions on a particular WiFi link and the frame transmissions on other WiFi links to compute the probability of frame

loss due to hidden interference [149]. In §4.3, we experiment with non-WiFi interferers operating alongside hidden terminals and show that WiFiNet is correctly able to identify the true interferer.

Interactions with external interference. External interference can be caused by non-WiFi devices that are not a part of the enterprise or by other non-enterprise wireless traffic (e.g., traffic from nearby WiFi networks). In both these cases, interference estimation can proceed as is if at least one of the APs is able to capture the pulse (or frame) transmissions from the interference source. However, if the transmissions from the non-WiFi device (or the external WiFi transmitter) are not captured by any WiFiNet AP, then WiFiNet controller would not be able to identify the source of interference.

Localizing a non-WiFi device instance

WiFiNet uses a computationally efficient, real-time localization scheme that imposes *zero* profiling overhead, and physically locates the non-WiFi device instance of *unknown transmit power* using a modeling based approach. Below, we explain our localization models.

Model-based localization

Let $\hat{\mathbf{r}} = [\hat{r}_0, \ldots, \hat{r}_{N-1}]$ be the mean RSS vector of all the pulses present in the cluster assigned to a non-WiFi device instance. For localization, we only consider the APs with valid received powers (*i.e.*, $\hat{r}_i \neq \phi$). We divide the entire region into grids of size 0.25×0.25 meters. Let i denote the grid location of AP_i. Let d_{ij} denote the distance between grids i and j. Let $P(l|\hat{\mathbf{r}})$ denote the probability of the non-WiFi device being at location l, given that the received power vector is $\hat{\mathbf{r}}$. We wish to determine the grid location l such that $P(l|\hat{\mathbf{r}})$ is maximized *i.e.*, we want $\text{argmax}_l P(l|\hat{\mathbf{r}})$. Using Bayes' theorem, $P(l|\hat{\mathbf{r}})$ can be written as $P(\hat{\mathbf{r}}|l).P(l)/P(\hat{\mathbf{r}})$. Assuming all locations are equi-probable, and since $P(\hat{\mathbf{r}})$ is constant for all l, we have $\text{argmax}_l P(l|\hat{\mathbf{r}}) = \text{argmax}_l P(\hat{\mathbf{r}}|l)$, which can be calculated as $\text{argmax}_l \prod_{i=1}^{N-1} P(\hat{r_i}|l)$ (assuming independence [166]). Put another way, the grid location l where the non-WiFi device is most likely present

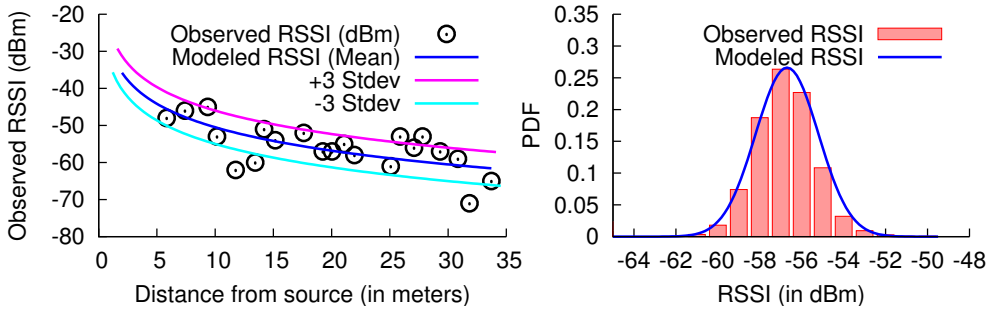

Figure 4.8: (left) Path loss model created by a WiFiNet APs using WiFi transmissions (right) PDF of actual RSSIs observed at a sample location and the model created using a normal distribution.

can be computed using,

$$\text{argmax}_l \sum_{i=1}^{N-1} \log P(\hat{r_i}|l) \tag{4.3}$$

Case of known transmit power (Model-TP). If the non-WiFi device instance is at a grid l, then the *expected* received power at AP_i (located at grid i) can be modeled as a normal distribution $\mathcal{N}(\mu_{il}, \sigma^2)$, where σ is the shadowing variable, and μ_{il} is the *expected mean* of the received power that can be modeled as $\mu_{il} = R^o - 10\gamma \log_{10} d_{il}$. Here, γ is the pathloss exponent and R^o is the power received from the non-WiFi device when placed at a distance of 1 meter from an AP (referred to as *transmit power*). How can we estimate γ for the non-WiFi device? WiFiNet APs derive γ using *WiFi frames i.e.*, each WiFiNet AP uses the data packets or beacons transmitted by neighboring WiFiNet APs to model the propagation loss characteristics (Figure 4.8) — since both WiFi devices and non-WiFi devices operate on the frequency, the propagation loss characteristics of their transmissions are similar. WiFiNet APs also compute σ^2, by measuring the variance in the received power values (Figure 4.8 (right)). Knowing μ_{il}, σ and $\hat{r}_i$, the controller can compute $P(\hat{r_i}|l)$ using,

$$P(\hat{r_i}|l) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(\hat{r}_i - \mu_{il})^2/2\sigma^2} \tag{4.4}$$

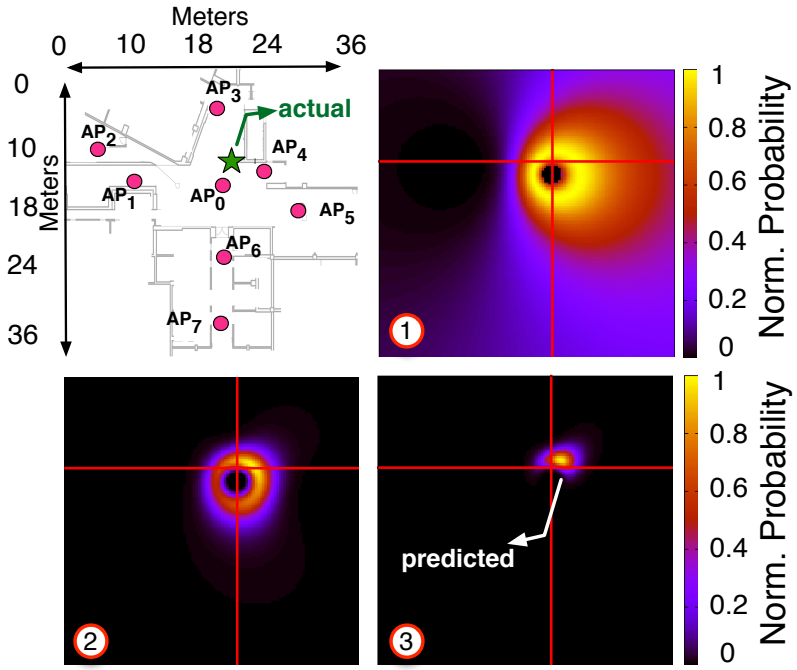

Figure 4.9: (top, left) Deployment 1 comprising 8 APs. (rest of the sub-figures) FHSS cordless phone device is placed at the starred location. Grid probabilities for predicted phone locations after processing 1, 6 and all AP pairs when using Model-UTP algorithm.

Intuitively, each AP_i propagates a probability that is maximum around a circle with center at grid i and radius equal to μ_{il}. If the transmit power R^o of the device is known, plugging in $P(\hat{r_i}|l)$ in Equation 4.3 and iterating over all the grids and APs, we can compute the grid l with the maximum probability of finding the device.

Case of unknown transmit power (Model-UTP). If R^o is not known, we can factor it out by considering each pair of APs: if the non-WiFi device is at a grid l, the *expected difference* in the mean received powers at AP_i and AP_j can be modeled as $\lambda(i,j,l) = \mu_{il} - \mu_{jl} = 10\gamma log_{10}(d_{jl}/d_{il})$, and expected

difference in the powers follows a normal distribution with twice the variance: $\mathcal{N}(\lambda(i,j,l), 2\sigma^2)$ [29]. Now, knowing $(\hat{r}_i - \hat{r}_j)$, we can compute $P((\hat{r}_i - \hat{r}_j)|l)$ as

$$P(\hat{r}_i, \hat{r}_j|l) = \frac{1}{2\sigma\sqrt{2\pi}} e^{-\left(\hat{r}_i - \hat{r}_j - \lambda(i,j,l)\right)^2 / 4\sigma^2} \tag{4.5}$$

and we can localize the non-WiFi device by finding

$$\text{argmax}_l \sum_{i,j} \log P\left((\hat{r}_i - \hat{r}_j)|l\right) \tag{4.6}$$

i.e., each AP pair propagates a probability $P((\hat{r}_i - \hat{r}_j)|l)$ on every grid l. The probabilities are high for the grids where the difference in received powers $(\hat{r}_i - \hat{r}_j)$ is close to $(\mu_{il} - \mu_{jl})$. After processing all AP pairs, the algorithm outputs the grid l with the maximum probability.

— *Example.* Figure 4.9 (top, left) shows a deployment of 8 APs along with the location of an FHSS cordless phone (shown using a star). The rest of the figures show how the grid probabilities indicating the location of the phone change after processing 1, 6 and all possible AP pairs.

Alternative localization methods

We also implemented several other localization schemes ranging from simple methods such as (i) *Strongest-AP*, picking the AP with the strongest received power as the device's location, and (ii) *Centroid*, picking the centroid of three APs with the strongest received powers, to more sophisticated approaches like (iii) an *Iterative* approach that performs an exhaustive search over all parameters (γ, σ, R^o, l) to find the grid l with the maximum probability, and (iv) a *Fingerprinting* approach where we profile the environment using *WiFi transmissions* — we transmitted WiFi packets using a laptop placed at several locations, and at each location WiFiNet APs measured the signal strength to derive the RSS vector. We then *normalize* the vectors to nullify the effect of a the transmit power of the laptop and derive a location's *fingerprint*. Localization is performed by measuring the RSS vector (after normalization) of a non-WiFi device and finding the fingerprint that is the closest match. In §5.7, we compare our model based localization algorithms to all the above methods.

The goal of our evaluation is to systematically benchmark WiFiNet's performance in diverse scenarios, and demonstrate its utility in realistic network settings. We break our evaluation into four parts: First, we demonstrate WiFiNet's ability in accurately characterizing the impact of different non-WiFi devices in a variety of scenarios. Second, we evaluate WiFiNet's accuracy in physically locating the non-WiFi interferers. Third, we emulate a non-WiFi interference prone enterprise WLAN scenario and show WiFiNet's utility in such a setting. Fourth, we benchmark different components of WiFiNet and highlight cases where WiFiNet's performance could degrade. We start by presenting the details of our implementation.

Implementation. We implemented WiFiNet using commodity WiFi APs equipped with Atheros AR9280 wireless cards that are connected to a central controller (Linux PC with 3.33 GHz dual core Pentium IV, 4 GB DRAM) over the Ethernet. Our implementation consists of few hundred lines of C code and 9800 lines of Python scripts that implement non-WiFi device detection functionality at the APs [131], perform synchronization across multiple APs, and implement clustering algorithms, interference analysis and device localization methods at the controller.

Evaluation set up. We experiment with devices in 2.4 GHz spectrum, and our current prototype has been tested with 5 different non-WiFi devices types : (i) high duty devices (analog cordless phones), (ii) fixed-frequency pulsed transmitters (ZigBee devices), (iii, iv) two types of frequency hopping devices (FHSS cordless phones, Bluetooth devices), and (v) broadband interferers (microwave ovens). We run our experiments on two different deployments: (i) Deployment 1 used 8 APs (Figure 4.9) and (ii) Deployment 2 used 4 APs (Figure 4.20). We experiment with different non-WiFi device locations, 802.11 rates, channel conditions and traffic patterns: (i) UDP with saturated traffic as well as reduced traffic loads, and (ii) replay of real HTTP/TCP wireless traces (§4.3). Unless otherwise stated, we run WiFi links on 802.11 rate to 6 Mbps and

use backlogged UDP traffic with a packet size of 1400B.

Ground truth. The conventional approach for measuring interference between WiFi links is to use bandwidth tests [71, 149] to determine the ground truth about the impact of a WiFi interferer on a WiFi link. We follow a similar approach to determine the impact of a non-WiFi interferer on a WiFi link: we perform controlled experiments where we send backlogged traffic on the WiFi link and (i) measure $p[L]$, the loss rate when the interferer is inactive, and (ii) measure $p[I \cup L]$, the loss rate when the interferer is active. However, unlike WiFi bandwidth tests wherein both the interferer and the link are using backlogged traffic and are active for the entire duration, a non-WiFi interferer may not be active all the time, leading to the case where $p[O]$ may be less than 1. For example, while high duty devices like analog cordless phones have $p[O] = 1$, other devices like FHSS cordless phones may only hop onto the WiFi channel of interest for a particular duration (when the WiFi link is backlogged, it turns out that $p[O] = 0.28$ for an FHSS cordless phone), or devices like microwave ovens have a characteristic duty cycle of 50% (*i.e.*, $p[O] = 0.5$). Therefore, impact given overlap, $p[I|O]$ has to computed as follows. $p[I \cup L]$ can be expressed as $p[(I \cup L)|O] \cdot p[O] + p[(I \cup L)|\neg O] \cdot p[\neg O]$. Now, $p[(I \cup L)|\neg O]$ is equal to $p[L]$, and expanding $p[(I \cup L)|O]$ in terms of $p[I|O]$ and $p[L]$, followed by a bit of algebra gives us

$$p^{\text{Actual}} = p[I|O] = \frac{\Big(p[I \cup L] - p[L]\Big)}{\Big((1 - p[L]) \cdot p[O]\Big)} \tag{4.7}$$

Using controlled experiments, we measure $p[L]$ (loss under isolation) and $p[I \cup L]$ (loss when the non-WiFi device is active). Now, knowing $p[O]$, we can measure $p[I|O]$ and subsequently compute the overall impact, $p[I]$. For experiments involving multiple devices, we measure the ground truth by activating only one device at a time and measuring its impact on the link. We note that the same ground truth ($p[I|O]$) is valid when multiple devices are activated simultaneously (the overall impact $p[I]$ may change, but $p[I|O]$

remains the same). WiFiNet, however, computes $p[I|O]$ estimates in presence of multiple, simultaneously active devices and WiFi links using any traffic load.

We further note that controlled experiments such as bandwidth tests require high overhead to compute the interference estimates for all links in the network. In operational networks, doing so may not be possible as such an approach requires network downtime where all the other WiFi links (and non-WiFi interferers) have to be silenced [149]. WiFiNet, on the other hand does not place any such requirements (e.g., backlogged traffic on links or network downtime), and can compute the estimates in real-time by passively accumulating frame and pulse transmission information.

Metrics used. For interference estimation, we compare WiFiNet's real-time, passive interference estimate of "impact given overlap" ($p[I|O]$) with that obtained using controlled experiments wherein the device is activated in isolation (ground truth). For localization, we report the difference in the actual and the predicted location of the non-WiFi device (*i.e.*, localization error) in meters.

Validating Interference Estimates

We start by validating WiFiNet's interference estimates across a variety of scenarios.

Single interferer scenarios

Method. We experiment with a total of 165 link-interferer scenarios comprising 4 non-WiFi devices — a microwave oven, an analog cordless phone, an FHSS cordless phone and a ZigBee transmitter. We activate each device in turn, and place it at different distances to vary the interference on the monitored WiFi link. We compute the ground truth (actual $p[I|O]$) using controlled experiments that measure the link loss rate when the device is active and that when the device is inactive. Next, we *randomly* activate and de-activate the non-WiFi device while the WiFi link is active and measure WiFiNet's real-time estimate.

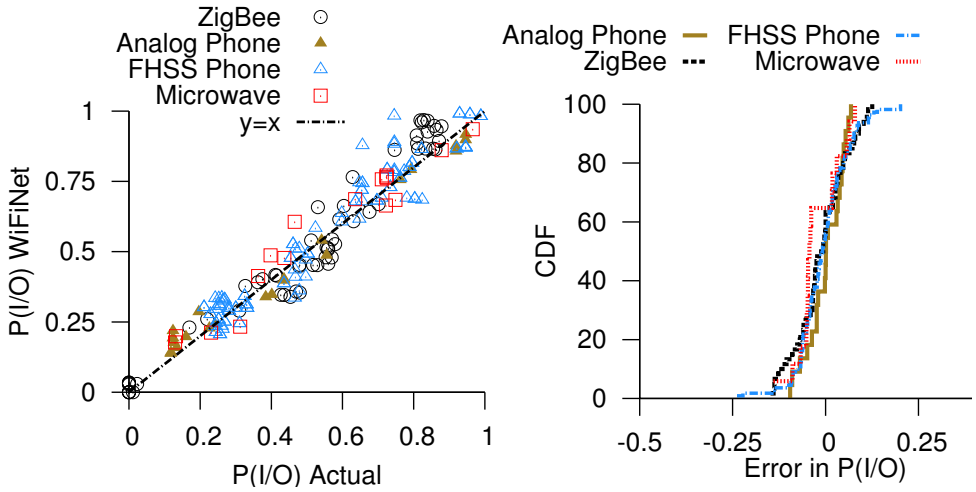

Figure 4.10: (left) Interference estimates obtained using controlled measurements (ground truth) and WiFiNet on 165 link-interferer scenarios comprising 4 different classes of devices. (right) CDF of error in interferer estimates is within ± 0.1 for 95% of the cases.

Results. Figure 4.10 (left) shows that WiFiNet correctly estimates a non-WiFi device's impact — across all device types and different amounts of interference (ranging from weak to strong), WiFiNet's estimates lie close to the ground truth (the points lie close to $y = x$). Figure 4.10 (right) shows that the overall error in WiFiNet's estimate is within ± 0.1 for more than 95% of the cases for all 4 devices.

Multiple interferers of different types

Method. In each run, we choose upto 4 random devices of different types, place them at random locations, randomly activate and de-activate them, creating scenarios when these devices are *simultaneously* active and measure WiFiNet's interference estimate for each device. For ground truth, we activate only one device at a time and perform controlled measurements. We repeat the experiments for different combinations of devices and locations.

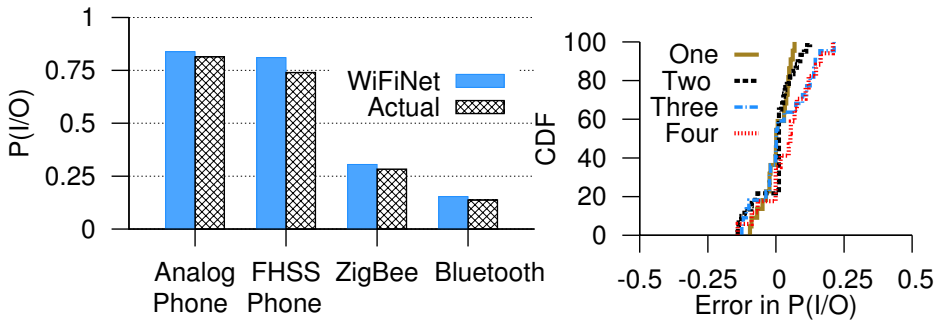

Figure 4.11: Accurately identifying impact of each interferer in the presence of multiple non-WiFi devices. (left) example scenario showing WiFiNet is able to identify the strong interferers (analog cordless phone, FHSS phone) and weak interferers (ZigBee and Bluetooth devices) accurately. (right) CDF of error in inteference estimates in the presence of multiple interferers.

Results. Figure 4.11 (left) shows a particular run which comprised two strong interferers (analog phone and FHSS cordless phone) and two weak interferers (ZigBee and Bluetooth devices). We find that WiFiNet is not only able to accurately identify the strong and weak inteferers, but is also able to discern the exact impact of each of these devices in spite of them being active simultaneously. Figure 4.11 (right) shows the CDF of error in interference estimates for combinations of 2, 3 and 4 devices across 60 runs. While the overall error slightly increases with increase in the number of devices, the error is within ± 0.15 for more than 85% of the cases even when operating 4 devices. The slight increase in error is due to increased overlap in the transmissions from multiple devices. We benchmark the effect of overlapping transmissions in §4.3.

Multiple interferers of the same type

Method. We now evaluate WiFiNet's performance when simultaneously operating multiple devices of the *same type*. We use 4 FHSS cordless phone devices — one base/handset pair is placed close to the WiFi link (to create

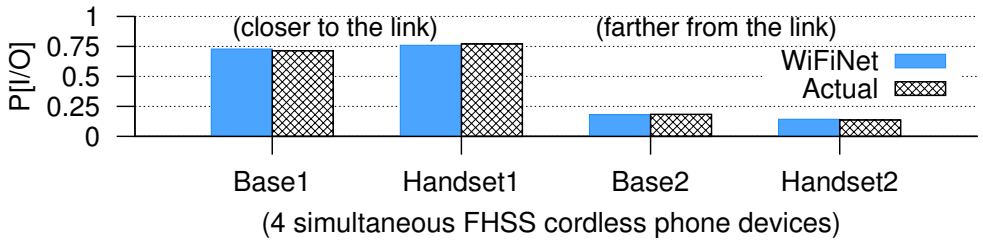

Figure 4.12: WiFiNet's accuracy in the presence of multiple non-WiFi devices of the same type. Out of 4 FHSS cordless phone devices, 2 are placed close to the link, and 2 are placed farther away.

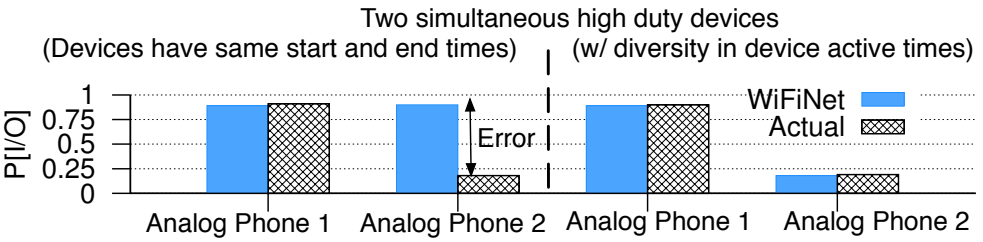

Figure 4.13: (left) Switching on and off 2 high duty devices (analog phones) *at the exact same time* causes WiFiNet to incorrectly identify the interferers. (right) Allowing diversity resolves the issue.

strong interference), whereas the other pair is placed farther away (to create weak interference).

Results. Figure 4.12 shows that WiFiNet is able to (i) accurately identify all 4 FHSS cordless phone devices using clustering mechanisms (benchmarked in §4.3) and (ii) accurately identify strong interferers (base/handset pair placed close to the link) and weak interferers (base/handset pair placed farther away from the link).

Case of multiple high duty devices

Method. We place an analog phone near the WiFi link (strong interferer) and another analog phone (operating at a frequency different from the first one), farther away from the WiFi link (weak interferer). We show results for two

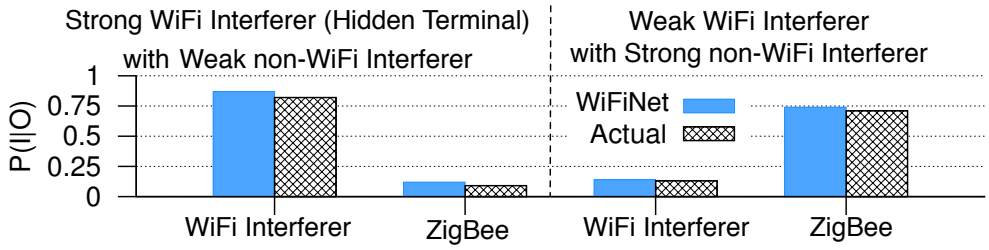

Figure 4.14: Estimating the interference impact of a WiFi interferer (hidden terminal) and a non-WiFi interferer (ZigBee device).

cases: (i) we switch activate and de-activate both the phones *at the exact same time*, and (ii) we activate the second phone 5 seconds after the first phone.
Results. Figure 4.13 (left) conveys that while WiFiNet is able to identify two different phones (by virtue of their different center frequencies), it incorrectly tags both the phones as strong interferers. Since both the analog phones are switched on and off at the *same time*, and they are high duty devices that are *always-on* (*i.e.*, they continuously emit energy), WiFiNet cannot distinguish between the interference impact of the two devices. Figure4.13 (right) shows a more practical scenario where introducing a little diversity (*i.e.*, a lag of 5 secs in the device start times) allows WiFiNet to correctly estimate the interference.

Mix of WiFi and non-WiFi interference

Method. We evaluate WiFiNet's accuracy when simultaneously operating a WiFi interferer (hidden terminal) and a non-WiFi interferer (ZigBee device). The interferers are placed at different distances from the monitored WiFi link to create two scenarios: (i) strong WiFi interferer with a weak non-WiFi interferer (ii) weak WiFi interferer with a strong non-WiFi interferer. The WiFi interferer's traffic follows an http on-off model for with sleep and active times derived from a wireless trace [137], whereas the ZigBee device used a constant bit rate. As before, to measure ground truth, we operate the devices in isolation.
Results. Figure 4.14 shows the results. In case (i), WiFiNet finds that losses are more likely to happen when the monitored link's frames overlap with WiFi

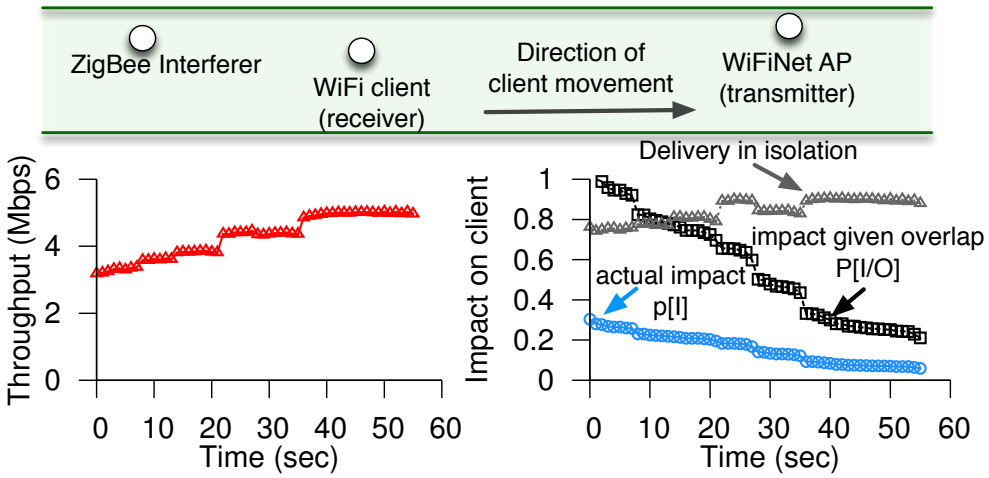

Figure 4.15: WiFiNet's ability to track the changing interference patterns for a client that is moving away from a ZigBee interferer. (left) instantaneous throughput at the client (right) delivery in isolation (*i.e.*, in absence of overlap), impact given overlap (p[I|O]) and actual impact (p[I]) are shown.

interferer's frames, whereas in case (ii), the losses show a high correlation when frames overlap with non-WiFi device's transmissions, resulting in accurate estimates for both cases.

Dynamic interference settings

Handling WiFi client mobility. We now evaluate WiFiNet's ability in updating the interference estimates that reflect the changing impact of a non-WiFi interferer due to client mobility. We use the set up shown in Figure 4.15 (top) where in a WiFi client is moving away from a ZigBee interferer. In the figure, plot on the left shows the instantaneous throughput at the client increases as it moves away from the interferer. The plot on the right shows WiFiNet's ability to track (i) delivery in isolation (*i.e.*, in the absence of overlap) that shows a slight increase, (ii) the impact given overlap p[I|O], which rapidly drops down from 0.98 to 0.2 as the client moves farther away and (iii) the actual impact p[I], owing to the probability of overlap, drops from 0.3 to 0.12. The decrease in the actual

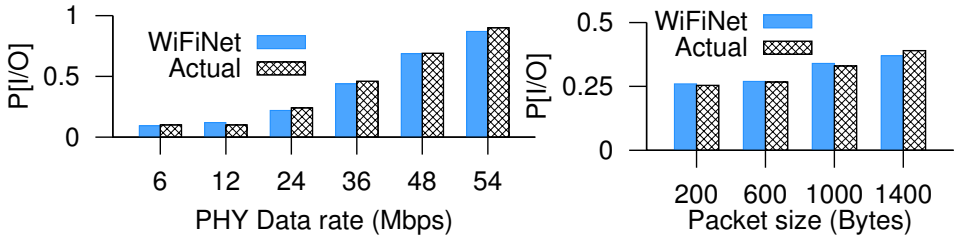

Figure 4.16: (i) Impact of PHY rate and (ii) packet size on p[I|O] in presence of a ZigBee interferer. For (i), packet size is fixed at 1400 bytes, and for (ii), rate is fixed at 12 Mbps. p[I|O] rises sharply with rate, the change in p[I|O] with packet size is less pronounced.

impact closely matches with the increase in throughput confirming WiFiNet's utility in understanding client performance in dynamic wireless environments.

Variable 802.11 rates and packet sizes. We evaluate WiFiNet's ability to dynamically track the changing interference estimates due to changes in (i) PHY rates and (ii) packet sizes used by the links. For ground truth, we perform controlled experiments at each PHY rate, whereas for WiFiNet we enable dynamic rate adaptation using SampleRate and capture the estimates in real-time. Figure 4.16 (left) shows that WiFiNet's estimates derived from rate adaptation closely match the ground truth. Since higher rates require higher SINR to decode a frame successfully, impact of the interferer *increases* with the increase in rate. Next, we fix the PHY rate (to 12 Mbps) and repeat our experiments for different packet sizes. Figure 4.16 (right) shows that WiFiNet is correctly able to track the slight increase in the interferer's impact at larger packet sizes.

Replay of wireless traces. We evaluate WiFiNet's performance using publicly available Sigcomm 2004 traffic traces [137]. We partitioned the trace into heavy, medium, and light periods corresponding to periods with airtime utilization of more than 50%, between $20-50$%, and less than 20% respectively, at different times of the conference [149]. The HTTP/TCP sessions are then replayed on

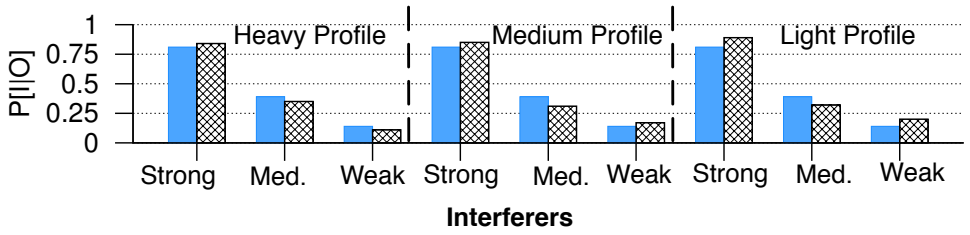

Figure 4.17: WiFi links replay real HTTP/TCP wireless traces (heavy, medium, and light profiles) in presence of strong, medium and weak interferers. WiFiNet's estimates closely match the ground truth in each case. The slight mismatch is due to the variability in packet sizes as the ground truth was measured using 1400 byte packets, whereas the traces comprised packets of different sizes.

Delay	**Min.**	**25th %ile.**	**median**	**75th %ile**	**Max.**
Convergence time	319 ms	549 ms	972 ms	1.7 sec	3.6 sec

Table 4.1: Distribution of convergence time for WiFi links replaying HTTP/TCP wireless traces (heavy, medium and light profiles) in presence of an FHSS cordless phone interferer.

WiFi links (using the mechanism described in [45]) in the presence of strong, medium and weak ZigBee interferers. Each client emulated the behavior of one real client from the trace, faithfully imitating its HTTP transactions. Figure 4.17 shows that that WiFiNet's interference estimates are close to that of the ground truth across different traffic profiles and interfering scenarios. The slight differences between the estimates are due to the variability in packet sizes in the real traces, compared to the ground truth that was measured using 1400 byte packets. We also show the CDF of time taken by WiFiNet to converge to the right p[I|O] estimates in Table 4.1 (median $<$ 1 sec). We benchmarks the factors affecting convergence time in §4.3.

Accuracy of Localization

We now evaluate our localization algorithms.

Algorithm	Min. error	25th %ile.	median	75th %ile	Max error
Iterative	0.3m	0.8m	2.1m	4m	10m
Model-TP	0.3m	0.3m	1.3m	3m	8m
Model-UTP	0.3m	0.8m	1.3m	4m	11m

Table 4.2: Overall localization error for an analog cordless phone and an FHSS phone when placed at random locations in deployment 2.

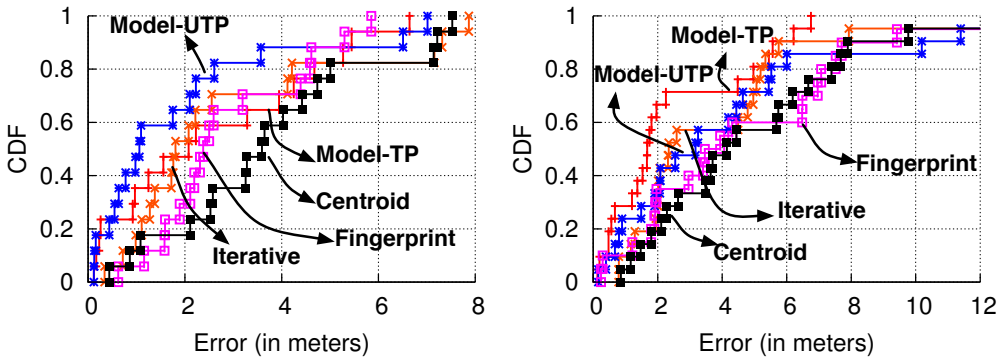

Figure 4.18: Accuracy of localization for (left) FHSS cordless phone and (right) analog cordless phone for deployment 1 (Figure. 4.9).

Scheme	Min. error	25th %ile.	median	75th %ile	Max error
Uniform γ	0.2m	1.9m	3.6m	7m	12m
per-AP γ	0.2m	2.0m	1.7m	2.3m	6.7m

Table 4.3: Overall localization error for the Model-TP algorithm with (i) uniform and (ii) per-AP path loss exponents (deployment 1).

Accuracy across different classes of devices

Figure 4.18 shows CDF of localization error for two non-WiFi device types: (i) frequency-hopping cordless phone and (ii) high duty, analog cordless phone, when using deployment 1 with 8 APs shown in Figure 4.9. Devices were placed at random locations and for each location, we compute the difference in the predicted and actual location for 5 different localization schemes (§4.2). We find that all algorithms perform well, resulting in a median error of 1–3 meters for

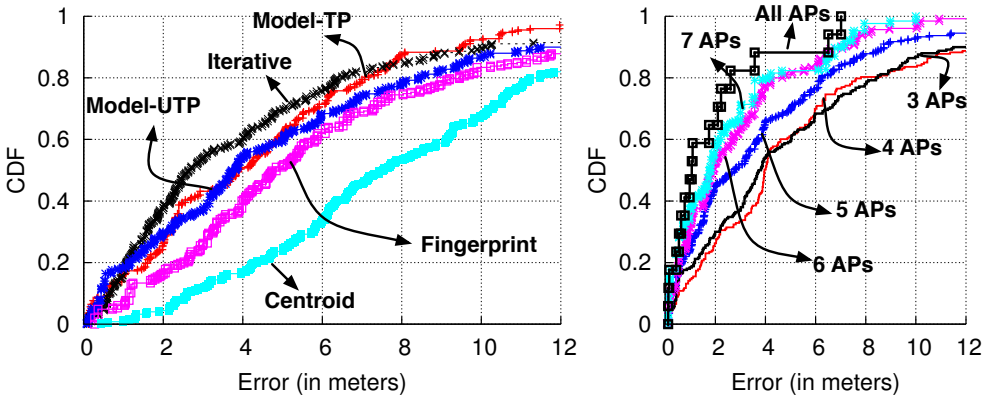

Figure 4.19: Localization accuracy for FHSS cordless phone (left) for subsets of 4 APs from deployment 1 (right) using Model-UTP when the number of APs was decreased from 8 to 3.

the FHSS phone, and 1.7–4 meters for the analog phone. Here, WiFiNet's modeling based localization approaches perform similar to the Iterative approach that employs an exhaustive search, and is better than Fingerprinting (§4.2) that incurs a profiling overhead. Accuracy of Fingerprinting, however, can be improved by increasing the density of fingerprints (0.05/sq.meter in this case) at the cost of a higher profiling overhead.

Effect of AP density

In each run, we randomly chose a subset of 4 APs (out of the 8 APs in deployment 1) and compute the localization error. We repeat the experiment for 25 runs and report the average error in Figure 4.19 (left). We observe that when the density of the AP deployment is sparse, the performance of Centroid algorithm worsens (median error of 8 meters) compared to the other algorithms (median error of 2.5 to 4.8 meters). Figure 4.19 (right) shows the degradation in the performance of Model-UTP, when the number of APs is reduced from 8 to 3. The median error only increases from 1 meter to 4 meters indicating the better performance of modeling based approaches in sparse deployments.

Improvements with fine-grained modeling

To understand the benefits from using a per-AP path loss exponent, we compare the performance of our modeling-based localization approaches when a uniform path loss exponent is used. Table 4.3 shows that when switching to a uniform path-loss exponent, the median error increased from 1.7 to 3.6 meters, and the maximum error increased from 6.7 meters to 12 meters. Using a per-AP path loss improves the WiFiNet's localization accuracy as it takes into account the differences in the environments surrounding the APs (e.g., walls and other obstacles).

Location insensitivity

We repeated our experiments to benchmark the performance of our algorithms in a different topology and environment (deployment 2 with 4 APs, Figure 4.20). Table 4.2 shows the overall error for the modeling-based and Iterative approaches. We find that the algorithms perform well with a median error of 1.3—2.1 meters.

Impact of transmit power

Our experiments with localizing Bluetooth devices resulted in an increased median error (2—6.7 meters) — owing to its low transmit power, only one of the APs could detect the Bluetooth device. In this case, the WiFiNet resorts to the Strongest AP approach for localization (§4.2).

Emulating an Enterprise WLAN

We now try to emulate the structure of our in-building WLAN by placing a WiFiNet AP near each production AP and distribute clients into offices (Figure 4.20). Our topology consists of 4 APs and 6 clients. We use a total of 9 non-WiFi interferers: 2 analog phones (high duty devices), 4 FHSS cordless phone devices, a Bluetooth device (frequency hopping devices), a ZigBee device (fixed frequency, pulsed transmitter) and a microwave oven (broadband interferer). WiFi links are assigned channels (shown in Figure 4.20) so as to

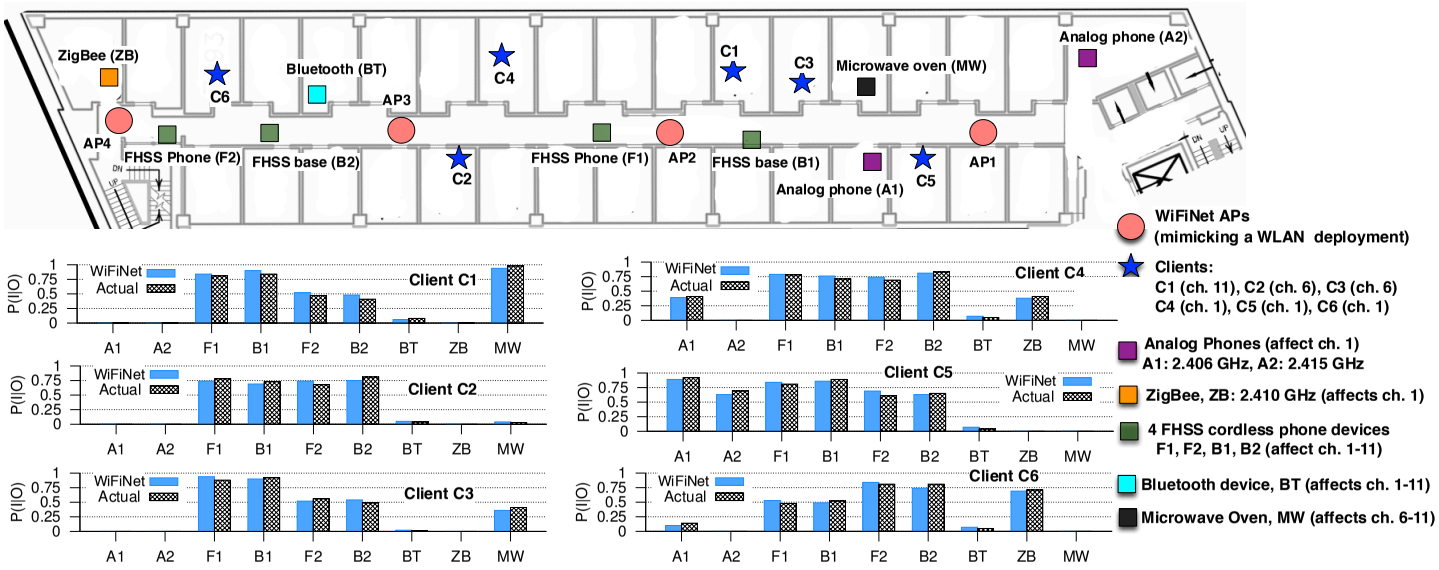

Figure 4.20: Emulating an enterprise WLAN with 4 APs and 6 clients. A total of 9 non-WiFi devices are placed to interfere with the clients: 2 analog phones, 4 FHSS cordless phone devices, a Bluetooth device, a ZigBee device and a microwave oven. WiFiNet is able to accurately characterize the interference impact (p[I|O]) of all devices (even those of the same type) on each of the clients.

create a scenario where each non-WiFi device affects at least one link. Each WiFi link follows an HTTP traffic model, with on-off times derived from a wireless trace [137]. We activate and de-activate the non-WiFi devices randomly, creating scenarios when devices are simultaneously active. As before, for ground truth measurements, we activate only one device at a time.

Figure 4.20 shows the interference impact of each interferer on the WiFi links — depending on the channel of operation, location of the client, and overlap probability (based on the actual WiFi traffic and non-WiFi device activity), WiFi links experience different amount of interference from each non-WiFi device. Further, WiFiNet's estimate *closely matches* the ground truth for each case. We find that all 4 FHSS cordless phone devices affect *all* the WiFi links ($p[I|O]$ varied from 0.45 to 0.8 due to their high transmit power of -20 dBm). The overall impact $p[I]$, however, only varied from 0.1 to 0.31 owing to their frequency hopping nature. Peak emissions of microwave ovens are typically in 2.45 to 2.47 GHz, and so the oven severely affected the client C1 which operated on channel 11. It is interesting to note that C3 (operating on channel 6) was also affected by the oven ($p[I|O]$=0.36) as it was close to the device, whereas C2 (channel 6, farther from the device) and C5 (channel 1, closer to the device) were not affected. Bluetooth device, due to its low power and adaptive frequency hopping nature did not significantly affect any of the links. On the other hand, high powered and high duty device like analog phones (A1 and A2) affected the clients on channel 1 (C4, C5, C6) much more than the ZigBee device that had a lower transmit power.

Microbenchmarks and Other results

We now benchmark convergence time, clustering algorithms, highlight cases where WiFiNet can under perform, and present results on estimating sender-side interference.

Convergence time

We define the convergence time as the time taken by WiFiNet to gather sufficient samples (*i.e.*, overlaps between WiFi frames and non-WiFi transmissions)

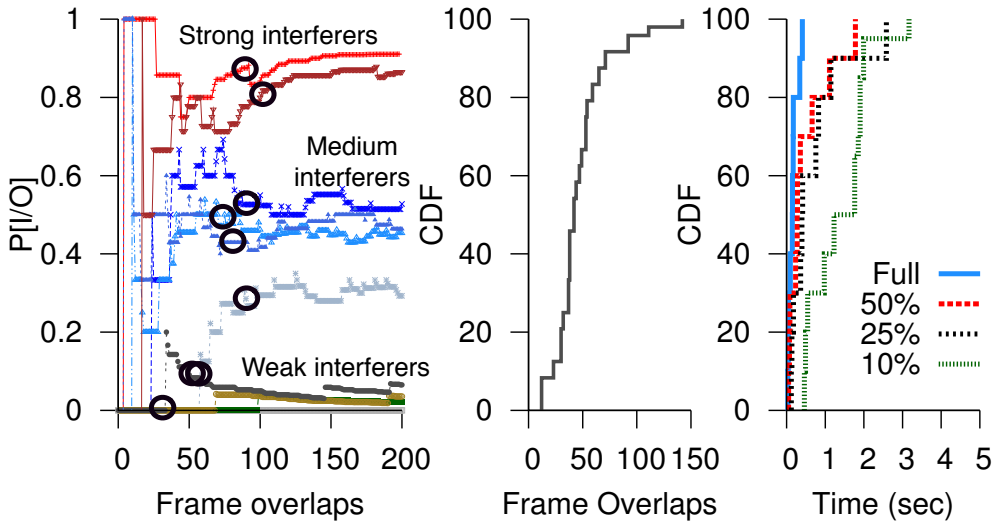

Figure 4.21: (left) Number of frame overlaps are required to converge for 10 ZigBee interferer scenarios including strong, medium and weak interference (middle) CDF of the number packet overlaps required for p[I|O] to converge (right) Convergence time as a function of the traffic load for an FHSS cordless phone.

to compute an accurate p[I|O] estimate (within ± 0.1 of the ground truth). Figure 4.21 (left) shows 9 different scenarios where a ZigBee interferer causing strong, medium or weak interference is activated along with a WiFi link. Across all scenarios, we find that < 100 overlaps between WiFi frames and ZigBee transmissions are enough for p[I|O] to converge (convergence points shown with black circles). Across different non-WiFi interferers and links < 150 overlaps are enough to converge to the ground truth (CDF shown in Figure 4.21 (middle)). The time for convergence depends on the WiFi link's traffic load, and the activity of the non-WiFi device. Figure 4.21 (right) shows that although the convergence time increases with lesser traffic, it is less than 4 seconds across a variety of traffic loads when using an FHSS cordless phone as an interferer. For devices like microwave ovens and analog cordless phones, convergence time was much lesser owing to increased overlaps.

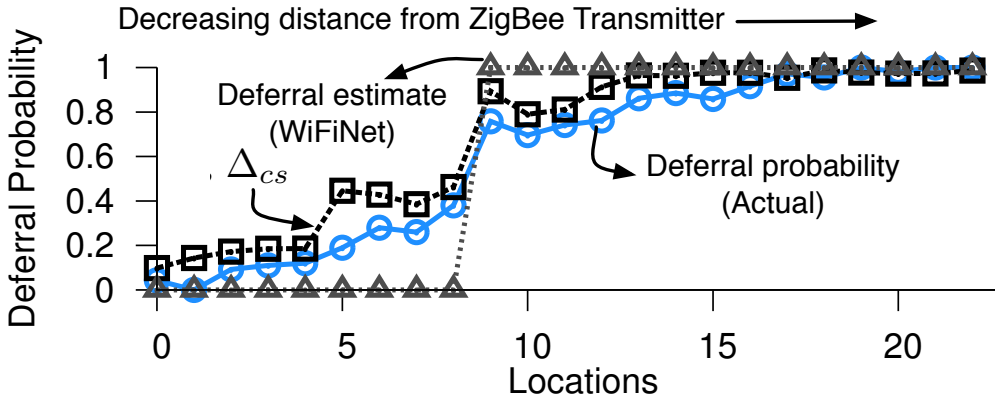

Figure 4.22: WiFiNet's estimates of deferral probability close match the ground truth. Here, a WiFi transmitter is moving toward a ZigBee interferer leading to increase in the deferral probability.

Estimating sender-side interference

We also benchmarked WiFiNet's ability to correctly estimate the carrier sensing interference across a number of non-WiFi devices and links. Here, we move a WiFi transmitter toward a ZigBee device (periodically transmits 4 ms pulses) and measure its deferral probability (§4.2). For ground truth, we measure the transmitter's sending rate when the device is active and that when the device is inactive. WiFiNet estimates the deferral probability in real-time — we observe that Δ_{cs} *i.e.*, the fraction of Case (2) instances (§4.2) increases as we move the transmitter away, indicating increased deferral. Further, Δ_{cs} also closesly matches the ground truth deferral probability.

Performance of clustering

Clustering is straightforward in many cases e.g., when the devices are of different types, or in the case of fixed-frequency devices (of the same type) using different center frequencies. We benchmarked our RSS and timing based clustering algorithms (§4.2) for the harder cases of (i) fixed-frequency devices using the same center frequency and (ii) frequency hopping devices. Table 4.4 shows the overall summary (when operating up to 4 devices of the same

Algorithm	Attribute	Clustering performance		
		% Correct	% Over-cluster	% Under-cluster
DBSCAN	Timing	92.7%	5%	2.3%
DBSCAN	RSS	88.7%	5.2%	6.1%
k-Means + EM	Timing	97.6%	1.3%	1.1%
k-Means + EM	RSS	91.4%	6.5%	2.1%

Table 4.4: Performance of clustering mechansisms used in WiFiNet. Results for two clustering algorithms (DBSCAN and k-means+EM) using (i) start time offset and (ii) RSS attributes are shown. Up to non-WiFi devices of the same type were placed at random locations.

type). We find that clustering algorithms perform reasonably well with $> 88\%$ accuracy in detecting the number of device instances. In case of over-clustering, the number of pulses in the extra clusters were relatively low, allowing us to discard the false positives. Under-clustering, however, can lead to error in estimates that can happen if the devices are close to each other (§4.3). Using timing attributes (when available) results in increased accuracy, compared to RSS based clustering, as timing attributes are not sensitive to the distance between devices (§4.3). Also, k-means+EM clustering has higher accuracy compared to density based clustering (DBSCAN).

Sources of error

We now highlight some of the scenarios where WiFiNet's performance can degrade.

Overlapping transmissions. We now benchmark the effect of transmission overlaps between multiple interferers. Figure 4.23 (left) shows WiFiNet's interference estimates in the presence of a strong and a weak non-WiFi interferer, as a function of the overlap between their transmission times. In the unlikely case when the transmissions from both non-WiFi devices overlap 100% of the time, WiFiNet is unable to distinguish between the two. However, as the percentage of overlap decreases, WiFiNet is able to discern the impact of the weak interferer. In practice, we expect diversity in device transmission

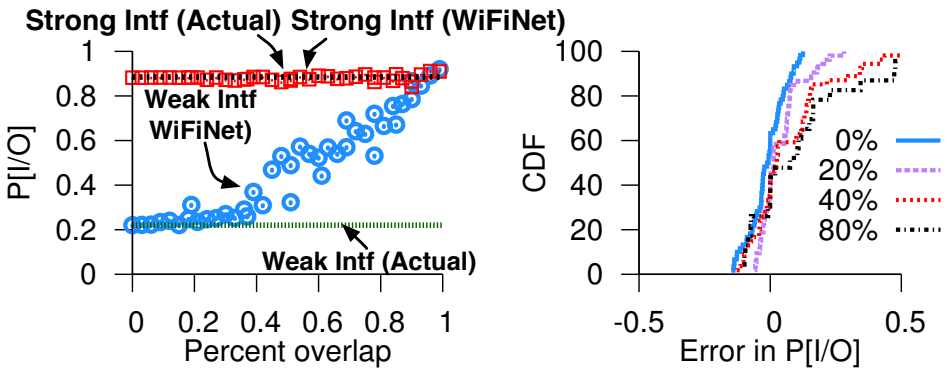

Figure 4.23: (left) Ability of WiFiNet to correctly identify interferers when transmissions from two non-WiFi devices overlap. p[I|O] measured by WiFiNet for both strong (p[I|O] = 0.88) and weak (p[I|O] = 0.22) interferers as a function of their overlap in transmission times. If the overlap is less than 45%, WiFiNet can distinguish the strong and weak interferers accurately. (right) Ability of WiFiNet to correctly estimate p[I|O] of an interferer as function of percentage of pulses lost (*i.e.*, not captured) by an WiFiNet AP.

times [131] to allow WiFiNet to output accurate interference estimates.

Coverage. WiFiNet's ability to derive an accurate interference estimate depends on how well the non-WiFi device's transmission are captured. In particular, p[I|O] and p[L] estimates will differ from the ground truth when none of the WiFiNet APs capture the device's transmissions. Figure 4.23 (right) shows the impact of losing transmissions from non-WiFi interferers — the error in estimates increase with decrease in the percentage of captured transmissions. In a typical enterprise deployment with multiple APs, this might not be a concern as we can expect at least one AP to capture the device's transmissions.

Proximity between devices. We now present a case when clustering mechanisms can under perform. We experiment with 2 FHSS cordless phone devices and place them at different distances. Figure 4.24 shows that timing based clustering is unaffected by the distance between the devices, but RSS based clustering is not. When the devices are placed $\leqslant$ 5 meters apart (at L1, L2), RSS

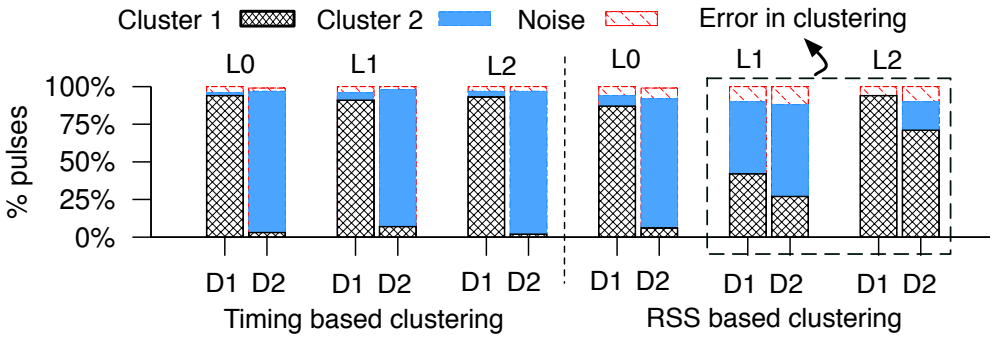

Figure 4.24: Performance of clustering for 2 FHSS cordless phone devices as a function of the distance between them. Clustering using (i) timing properties is unaffected by the distance, whereas that using (ii) RSS performs incorrect clustering when the devices are placed $\leqslant 5$ meters apart (L1, L2). For L0, devices were 10 m apart.

based clustering cannot distinguish them as their RSS vectors look similar. This results in under-clustering (pulses of both devices are put into one cluster) or incorrect clustering (each cluster has a mixture of pulses from both devices).

4.4 ISSUES AND DISCUSSION

Our work on WiFiNet is a first step towards estimating non-WiFi interference using commodity WiFi hardware in today's WLANs. We now comment on some of the limitations and discuss the scope for improvements.

- **Localization accuracy.** While WiFiNet's localization accuracy is reasonable (a median error of $\leqslant 4$ meters), the worst case error goes up to 15 meters. We believe that for purposes of non-WiFi device localization, this worst case error is tolerable. For example, in most cases, such an error would still be able to localize the non-WiFi device to within a room, or an office in a large enterprise. We note that accuracy of our localization mechanisms can be further improved in a number of ways such as increasing the density of APs [112], utilizing more complex

propagation models [65, 75, 120], using richer information about the deployment [75] and exploring newer localization algorithms [41].

- **Handling mobile non-WiFi devices.** While WiFiNet can handle client mobility, we did not experiment with the case where non-WiFi devices themselves are mobile. We note that WiFiNet should be able to handle mobile non-WiFi devices such as analog phones, or wireless video cameras, which have detection times of less than 100 ms (Chapter 3). However, localization accuracy for frequency hopping devices such as Bluetooth or frequency hopping cordless phones would be affected because Airshark's current detection times for these devices are in the order of seconds. Further, mobile non-WiFi device localization would also be problematic in the case of multiple devices of the same type that do not exhibit any device-specific characteristics (e.g., two bluetooth devices). This is because, in such cases, received power information at the APs is essentially used as an identifier in our clustering algorithms (Chapter 4).
- **Overlapping transmissions.** Currently, WiFiNet cannot handle the case where transmissions from both non-WiFi devices overlap most of the time. In such cases, the difficulty arises as it is not clear which device is responsible for the observed interference. We note that as the percentage of overlap decreases, WiFiNet is able to discern the impact interferers correctly. In practice, we expect diversity in device transmission times [131] to allow WiFiNet to output accurate interference estimates.

4.5 SUMMARY OF WIFINET

In this chapter, we presented WiFiNet, a system to estimate the interference experienced by WiFi links in presence of non-WiFi devices using only WiFi hardware. WiFiNet can correctly estimate the impact of each non-WiFi device, in presence of multiple other interferers, even if they are of the same type. It also correctly tracks changes due to client mobility, dynamic traffic loads, and varying channel conditions. Further, WiFiNet also identifies the physical

locations of non-WiFi devices. We believe a system such as WiFiNet can help WLAN administrators use commodity WiFi APs to better understand and manage non-WiFi interference, especially in environments such as enterprise WLANs.

5 MODELING INTERFERENCE DUE TO FLEXIBLE CHANNELS

5.1 MOTIVATION

In previous chapters, we presented systems that help understand the impact of wireless interference using *post-mortem* analysis of wireless packet losses *i.e.*, the impact of interference is estimated by passively observing the event of a packet loss. We now shift our attention to alternative techniques using modeling procedures that help estimate the impact of interference before the event of a packet loss. There are variety of modeling mechanisms that have been explored to estimate the impact of interference between two WiFi links based on the signal strength relationships between these links [84, 133]. We adopt a similar approach in this chapter and try to understand the impact of such interference between WiFi links. Specifically, we model the interference between WiFi links that use different channel widths, and devise mechanisms to mitigate WiFi to WiFi interference using efficient channelization mechanisms. We defer similar modeling of non-WiFi interference scenarios for future work (Chapter 7).

Traditionally, wireless channels strictly correspond to a pre-defined center frequency and a specific channel width. It is well known that interference in such fixed-width setting can be modeled using signal strength relationship between the links [133]. While this strict notion of a fixed-width channels have proven to be useful, researchers in recent years have realized that flexible channels — channels in which the center frequency and bandwidth are picked based on traffic demands, noise and interference levels across a spectral band — can further improve spectrum efficiency. In the context of dynamic spectrum access networks and cognitive wireless networks, a large body of work [30, 32, 97, 152, 164, 168] has examined strategies to assign flexible channels. More recently, this problem of choosing the right frequency and width for communication has gained relevance with the onset of white-space networking where agile adaptation of these parameters is essential [24]. In this chapter, we explore modeling mechanisms that help us derive the interference relationship between

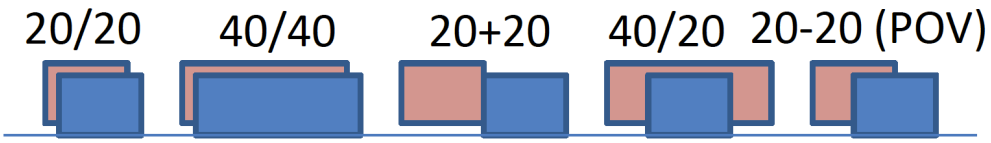

Figure 5.1: Example flexible channel configurations using two channel widths of 20 and 40 MHz. The total available spectrum is 40 MHz.

links using such flexible channels. Specifically, we focus on modeling the interaction between flexible channels in the context of 802.11-based networks.

Current 802.11 hardware can provide a limited amount of software-level flexibility that allows transceivers to operate on such flexible channels, e.g., a fixed number of channel widths (5, 10, 20, and 40 MHz) and a set of permissible center frequencies in the 2.4 GHz or 5 GHz band [67]. Using this flexibility, the work in [36] shows how a single 802.11 link can pick an efficient channel width to adequately meet its traffic demand. At a high level, [36] shows that increasing channel width for a *single, isolated* link potentially allows greater throughput. But that, for a given total transmit power used by a wireless card, the power per unit frequency reduces for larger widths [36], leading to reduced SNR and poor connectivity in longer links.

While the work in [36] focused on how to adapt the channel width for a *single, isolated link*, we focus on how to employ flexible channelization when using *multiple, potentially interfering links*. We look at the use of flexible channelization in a fairly complex and realistic setting — modeling the interference between flexible channels and improving throughput for an 802.11 enterprise WLAN using off the shelf hardware. The core problem we address in this chapter is the following:

> *"Given an enterprise WLAN with many different Access Points (APs) and arbitrarily located wireless clients, how should flexible channels for each AP be structured?"*

Initially, we imagined that the problem has an easy solution: identify the traffic demand for each AP (aggregated over all its clients) and provide a single

channel to each AP that is proportional to this traffic demand. The channel choices can be periodically adapted based on demand evolution. Indeed, work in [110] proposes and shows the benefits of such a solution through careful simulation based studies. However, in our attempt to implement such a solution on an 802.11 testbed, we quickly uncovered new challenges.

One of the biggest challenges was to create an effective model for a *conflict graph* — a graph that captures the interference between a link and a potential interferer. Prior work (e.g., [110, 164, 168]) assumes that the interference behavior of two, potentially conflicting, links is unaffected by changes in their channel widths. However, in reality, the interference properties of two links can be greatly impacted by their channel width of operation, even if they use the same channel configuration (i.e., the same width and center frequency).

We illustrate this through a simple, yet interesting example. Given two links and a spectrum band, say 40 MHz, there are many ways to assign flexible channels (Figure 5.1). Some natural choices are: (i) both links operate using the entire 40 MHz channel and time-share using regular random access mechanisms (**40/40** in Figure 5.1), and (ii) both links operate on separate 20 MHz channels (**20+20**) and potentially suffer no interference from each other. Initially, we assumed that examining these two choices alone is adequate to find the most efficient channel assignment. However, in our testbed experiments we found multiple two-link conflict scenarios where the best channel configurations were fairly non-standard, including: (iii) one link on a 40 MHz channel, the other on a 20 MHz channel, both with the same center frequency (**40/20**), (iv) both links on partially overlapped 20 MHz channels, **20-20(POV)**. Interestingly, we also found several cases where using a single 20 MHz channel (**20/20**) provided better throughput than operating the links on a single 40 MHz channel (40/40).

The reason these other channel choices proved to be the best configuration for some link topologies was due to the variable nature of conflict that changes with channel width, even when the center frequency of the two links is identical. In fact, through experiments we found that changing channel widths has a great impact on all wireless interference parameters, e.g., carrier sense and interference range, hidden terminals, exposed terminals, etc. There were many instances where two neighboring links were in carrier sense range when using

the same 20 MHz, but turned into hidden terminals when their channel widths were identically increased to 40 MHz. Exposed terminal scenarios sometimes appeared when reducing channel widths. More complex interference patterns arose in the presence of multiple links, and when considering different center frequencies, since some of the assignments resulted in partial spectral overlaps.

Hence, in our overall problem of assigning flexible channels in an enterprise WLAN, we have to compute the conflict graph *for all possible channel widths and center frequencies*. For an N node network using $|w|$ possible channel widths and k PHY data rates (e.g., for 802.11a, $k = 8$), this can require $O(N^2 \cdot k \cdot |w| \cdot 2^{|w+1|})$ measurements, one for each link pair, data rate, channel width, and center frequency (§5.4). This is a particularly daunting and complex task. To address this, we develop techniques to model the conflict graph using only $O(N \cdot k)$ empirical measurements at a *single* channel width. The next step is to use this conflict graph to assign flexible channels. In our proposed system, FLUID, a central controller improves the network throughput by assigning the center frequencies and widths to the APs on the fly, depending on the actual traffic demand. To further maximize the number of simultaneous transmissions, FLUID explores a joint data scheduling and flexible channelization approach. As we show in §5.5, the search space in this context grows exponentially in the number of transmissions. To tackle this, we propose a randomized algorithm with relatively low overhead to derive efficient transmission schedules, as demonstrated in our experiments.

We implemented FLUID on Atheros wireless cards running the MadWiFi driver [8] and have deployed the system on a 50 node testbed spanning multiple floors in our university building. Testbed results show that FLUID improves the median throughput by 59% across all possible PHY rates and when using dynamic rate adaptation, in a network-wide setting, compared to an approach using fixed width channels. To the best of our knowledge, FLUID is the first realization of an 802.11 based WLAN system consisting of multiple APs that are capable of operating at variable channel widths.

We make the following contributions in this chapter:

- We show that while flexible channelization can improve system throughput, its benefits in a network-wide setting are not immediate — careful construction of flexible channels requires taking into account the interference parameters like carrier sensing, hidden terminals etc., which *depend* on the combinations of frequencies and channel widths used, as well as the specifics of topology and traffic demand (§5.2).

- We develop a modeling framework to efficiently compute the conflict graph for an N node network employing flexible channelization using only $O(N.k)$ empirical measurements at a *single* channel width, as opposed to brute force approaches, which require $O(N^2.k.|w|.2^{|w+1|})$ measurements (§5.4).

- We present an algorithm to construct flexible channels, and show that combining flexible channelization with data scheduling can further improve network throughput (§5.5).

- Through a real deployment on our testbed, we evaluate FLUID over a variety of scenarios, and show that it can significantly improve the performance of a WLAN (§5.7).

5.2 PROPERTIES OF FLEXIBLE CHANNELS

Prior experimental work has noted three properties of varying channel widths on a *single, isolated* link [36]: (i) throughput of a link is proportional to the channel width, (ii) halving the channel width doubles the power per Hertz, and consequently increases the range by 3 dB[1], and (iii) reducing the width by reducing the clock rate (and hence sub-carrier spacing) results in lower battery consumption. One would expect the first two properties, in particular, link throughput, to be impacted by the interference from the other links in the network. In the rest of this section, we show that this is indeed the case and investigate the reasons behind this. Additionally, we show why designing a

[1]In Sec. **??**, we discuss how our models can be modified to work in systems where this property might not hold

network that uses flexible channelization presents new challenges.

Measurement methodology. We perform measurements on a 50 node testbed deployed across five floors of a building. Each node runs Linux 2.6.20 kernel and is equipped with two Atheros 5212 based 802.11 NICs. Modifications to the MadWiFi driver allowed us to write to the hardware register that configures the PLL, giving us the capability to use four channel widths of 5, 10, 20 and 40 MHz. We also made modifications to 802.11 timing parameters to ensure fair contention among different widths [36] . Experiments were carried out using 802.11a to avoid any external interference from our department WLAN that operates on 802.11 b/g. We experimented with *dynamic rate adaptation* and with *all fixed PHY data rates* i.e., 6 Mbps to 54 Mbps in the 802.11a system. Due to space constraints, we typically present a snapshot of results, often using three fixed PHY rate scenarios (12, 36, and 54 Mbps[2]), as well as when the SampleRate algorithm [8] is used to dynamically adapt the PHY rate across all possible 802.11a rates. For bandwidth tests, the nodes broadcast 1400 byte UDP packets at full sending rate for 10 seconds and experiments are repeated for 30 runs.

Impact of flexible channels

We observed that, in isolation, the throughput for high SNR links nearly doubles on doubling the channel width. However, in the presence of even one interferer, this property no longer holds. To show this, we randomly picked a 40 MHz interferer, and measured how the throughput of a randomly chosen good quality link (delivery ratio > 0.99) changes when it switches from 20 MHz to any of the other widths. The interferer and the link used the same center frequency. Table 5.1 shows the throughputs obtained at 40 MHz and 10 MHz (throughput normalized w.r.t. 20 MHz) for four different PHY rates and rate adaptation (SampleRate), across 2872 link/interferer combinations. Since the transmitter might be outside the interference range of a link, we observe that

[2]The data rate notations used in the thesis correspond to the PHY rates when the channel width is set to 20 MHz (the default in 802.11) . For e.g., 6 Mbps refers to OFDM with BPSK and coding rate of 1/2. The actual data rate would be doubled (or halved) when the channel width is set to 40 (or 10) MHz.

	20 ⟶ 10 MHz % links w/ Norm. Thr.			20 ⟶ 40 MHz % links w/ Norm. Thr.		
PHY Rate	0.5×	0.5×—1×	⩾ 1×	2×	1×—2×	< 1×
Fixed 6 Mbps	44%	41%	15%	31%	45%	24%
Fixed 12 Mbps	42%	45%	13%	29%	48%	23%
Fixed 36 Mbps	37%	44%	19%	24%	49%	27%
Fixed 54 Mbps	38%	41%	21%	20%	51%	29%
SampleRate	38%	39%	23%	27%	45%	28%

Table 5.1: Choosing the right width is non-trivial as throughput may not be proportional to channel width under interference. Plot shows UDP throughputs for 10 and 40 MHz widths (throughputs normalized w.r.t. 20 MHz) across 2872 link/interferer combinations for different fixed PHY rates and for dynamic rate adaptation (SampleRate). Shaded portion indicates the percentage of links for which the throughput is doubled (halved) when the width is doubled (halved).

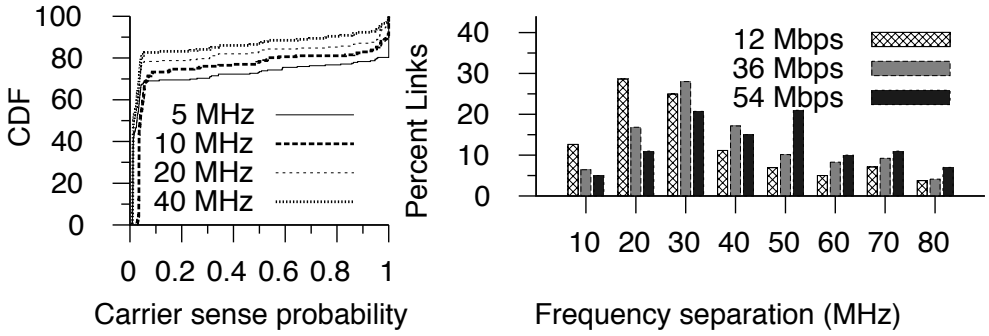

Figure 5.2: (left) Carrier sensing probability at different widths for 600 link pairs (right) Frequency separation needed for conflicting 40 MHz links to become non-conflicting at different PHY rates.

throughput doubles on doubling width for a certain fraction of the links e.g., when using rate adaptation, 27% of the links doubled the throughput when switched from 20 MHz to 40 MHz. However, this property doesn't hold in most other cases (unshaded portions in the table). For 45% of the links, the throughput *increases by varying amounts*—1× to 2×, and for the remaining 28% of the links, the throughput *decreases* when switched to 40 MHz. This holds

PHY Rate	Scenario	5 MHz	10 MHz	20 MHz	40 MHz
Fixed 12 Mbps	Hidden	78	127	81	46
	Exposed	139	114	117	149
Fixed 36 Mbps	Hidden	81	135	87	58
	Exposed	121	108	84	96
Fixed 54 Mbps	Hidden	96	141	97	74
	Exposed	107	99	73	69

Table 5.2: Number of hidden and exposed links depend on the channel widths. The precise methodology to identify hidden and exposed links was taken from [148].

for other widths as well, even though at varying degrees. In order to isolate the effect of PHY rate, we repeated the experiments across different fixed PHY rates and observed similar results (Table 5.1). To study why this happens, we look at the impact of widths on: carrier sense range, hidden and exposed links.

Carrier sensing range: Since smaller widths have higher energy per Hertz [36], we observed that more links carrier sense each other at lower widths. Figure 5.2 (left) presents the CDF of carrier sensing probabilities among 600 link pairs in our testbed for different widths. Around 33% of link pairs carrier sense each other at 5 MHz, while only 15% of link pairs carrier sense each other at 40 MHz.

Hidden and exposed links: Table 5.2 shows the number of hidden and exposed links at different channel widths (and rates). While we find that the number of hidden and exposed links *vary with widths*, there is no particular trend. This is because lower widths not only cause more links to be in carrier sensing range, but also interfere over longer distances.

Partial spectrum overlaps: The extent of interference between links also depends on the amount of spectral overlap [106]. In case of flexible channels, partial spectral overlaps can occur when links use same center frequencies but different channel widths, or if links operate at different center frequencies. Such an interaction between the links has to be well understood in order to

assign channels efficiently. For example, we observed that varying amount of frequency separation is needed between two conflicting 40 MHz links to make them non-interfering. Figure 5.2 (right) shows this for 279 link pairs when using different PHY rates. At 36 Mbps, only 17% of the link pairs require a separation of 40 MHz. 51% require less separation (offering the opportunity for spectrum reuse). The remaining 32% require more than 40 MHz of separation, implying that naively packing these links at a separation of 40 MHz can degrade throughput.

Constructing flexible channels

We now study the impact of the above properties when assigning spectrum to links in a network. To begin with, we ask a simple question: *"If a total of* 40 *MHz of spectrum is available, how should we assign it to two links?,"* and show that the solution has many interesting considerations. The best frequency and width assignment, changes depending on the topology and the interference among links.

Figure 5.3 shows throughput measurements for five simple two-link topologies taken from real instances in our testbed along with the five example spectrum assignments described in §5.1. Here, the configuration 40/20 refers to the link (t_1-r_1) operating on the entire 40 MHz and (t_2-r_2) operating on 20 MHz, using the same center frequency. The configuration 20-20(POV) refers to the partial overlap case where the two links use two 20 MHz channels with center frequencies separated by 10 MHz. A rounded rectangle enclosing two nodes represents a conflict (i.e., carrier sensing when the nodes are both transmitting, and interference when one node is transmitting and the other is receiving). The first column shows the topology information, while the rest of the columns illustrate how the *conflicts change* across different assignments and result in different throughputs. The throughput values (U) are normalized w.r.t. to the lowest throughput for each case. For the cases that require channel/width switching (case E1, 20+20, 40/20 and 20-20 (POV)) we use optimizations to reduce the switching overhead (§5.6). In all measurements, the traffic was

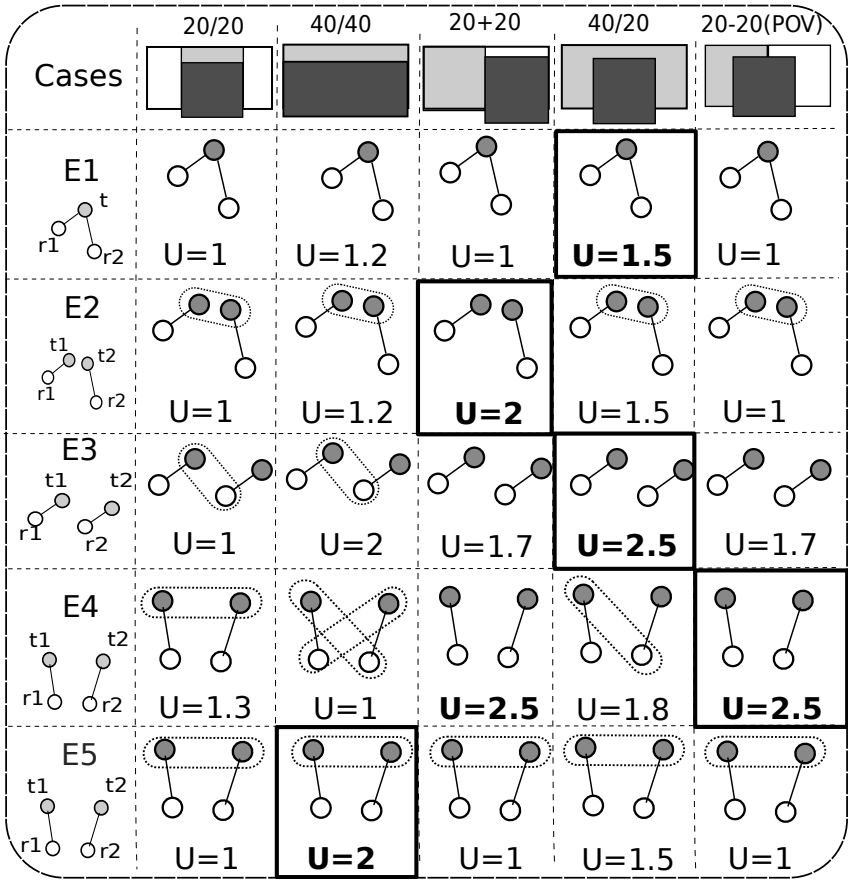

Figure 5.3: Conflict information and corresponding throughputs with different spectrum assignments for real topologies in our testbed. A rounded rectangle enclosing two nodes represents a conflict (i.e., carrier sensing when the nodes are both transmitting, and interference when one is transmitting and the other node is receiving).

backlogged on both links. For ease of exposition, in this section we present the results when the 802.11 PHY was set to the base rate of 6 Mbps.

We now briefly explain why the best spectrum assignment (shown in bold squares) differs in each case. **Case E1** in Figure 5.3 corresponds to the scenario where client r_2 has a low SNR and thus a poor delivery ratio at 40 MHz; the delivery ratio increases to 1 at 20 MHz because of 3 dB increase in SNR. For

client r_1, the delivery ratio is 1 at both widths. Here, using *client-centric* widths (40/20 in Figure 5.3) achieves the best throughput (a gain of 25% over 40/40). All other configurations have worse throughputs as they either waste spectrum or result in a poor delivery ratio for r_2.

We consider two links in **Case E2**, with link (t_2-r_2) having a poor delivery ratio at 40 MHz. Using 40/20 improves the delivery for r_2 (due to increase in SNR of the link) compared to 40/40. However, the links still continue to carrier sense because of which they cannot effectively use the entire 40 MHz spectrum. Here, 20+20 achieves a better throughput (a gain of 33% over 40/20) as both links can simultaneously operate on separate 20 MHz channels with good delivery ratios.

Case E3 illustrates the scenario of a one-way hidden terminal (t_1 interferes with r_2) which is resolved by separating the links on two 20 MHz channels (20+20). However, simply *narrowing the width resolves the conflict* — operating the link (t_2-r_2) at 20 MHz improves the SINR and hence makes the links non-conflicting. 40/20 improves the throughput by 47% over 20+20 due to increased transmission concurrency.

In a two-way hidden terminal scenario (**Case E4**), the best configurations resolve the conflict between two links, either 20+20, or partially overlapping assignment, 20-20(POV). Using 20-20(POV) might be more preferable for larger network scenarios as it uses lesser spectrum. Interestingly, using a single 20 MHz channel for both links (20/20) provides a better throughput than using a single 40 MHz channel (40/40), as the links carrier sense each other in the 20/20 configuration due to increase in their signal strengths.

Finally, **Case E5** represents a scenario where the links carrier sense each other when using the 20/20 configuration. However, it turns out that the links continue to carrier sense at all other configurations. For example, a center frequency separation of 20 MHz (20+20) is not adequate to resolve the carrier sensing conflict. Given this scenario, sharing the medium using 40/40 turns out to be the best configuration.

Why is the best flexible channel configuration different in each case? While the above is by no means an exhaustive set of flexible channel configurations, even exploring this limited set of configurations allowed us to observe cases

where different configurations performed the best. This is because conflicts (carrier sensing and interference relationships) between the links are determined by the flexible channel configuration used. These conflicts in turn determine the total throughput in each case. Thus, in each case we have to choose a configuration that minimizes the conflict and improves the overall throughput. Put another way, to maximize the overall network throughput, we have to employ a *conflict-aware* mechanism which intelligently chooses a particular flexible channel configuration based on the carrier sensing and interference relationships at different widths and center frequencies.

5.3 FLUID: OVERVIEW

We propose FLUID, a system that improves the wireless network throughput through the use of flexible channels. While the design of FLUID is generic and can be applied to any 802.11 based setting, in this work, we focus on its application to an enterprise WLAN setting.

Target network setting. Consider an enterprise WLAN setting where clients and APs are capable of operating on flexible channels. All the APs are connected over an Ethernet backplane, and are managed using a central controller. Let B be the total amount of spectrum in use. Let $|w|$ denote the total number of channel widths to choose from. Let w_{min} denote the minimum channel width used, and assume that channel widths are of the form $w{=}w_{min}{\cdot}2^r$, where $0 \leqslant r \leqslant |w| - 1$. In our implementation, $|w| = 4$ and $w_{min} = 5$, as we use 5 MHz, 10 MHz, 20 MHz, and 40 MHz as the possible channel widths. Let $\mathcal{F} = \{(f_c, w)\}$ be the set of permissible center frequency and width combinations s.t. f_c is of the form $f_c = w_{min}{\cdot}c$, where c is an integer and $[f_c - \frac{w}{2}, f_c + \frac{w}{2}] \subset [0, B]$.

We now sketch the main operations of FLUID. Figure 5.4 illustrates the different components involved in FLUID.

1. Conflict graph generation. FLUID builds a conflict graph to model the interference between links while taking into account the combination of channel

widths and center frequencies. Using a brute force approach for conflict graph computation becomes infeasible as it requires $O(N^2 \cdot k \cdot |w| \cdot 2^{|w+1|})$ measurements. As discussed in §5.4, FLUID uses modeling techniques to reduce the overhead to $O(N \cdot k)$.

2. Interference mitigation. The controller uses the conflict graph to mitigate interference and improve system throughput either by employing (i) an unscheduled approach i.e., flexible channelization with DCF or (ii) flexible channelization along with a scheduled approach such as CENTAUR [148], which can improve downlink performance. While we have explored both the approaches, in this thesis, we focus on the harder problem of improving downlink system throughput using a *joint scheduling and flexible channelization* approach. Although designing such a scheduled system is more challenging than its unscheduled counterpart, it offers better performance than DCF with static channel assignment mechanisms for the following reasons: (i) it uses spectrum efficiently as it takes the actual traffic into consideration, (ii) it resolves downlink hidden interference and opportunistically capitalizes on exposed terminal scenarios, (iii) using a scheduled approach enables an AP in FLUID to employ *client-centric* widths which is otherwise difficult to manage with DCF in the presence of upload traffic. In §5.7, we show that FLUID's scheduled approach performs better than CENTAUR and the unscheduled approaches across various scenarios.

5.4 MODELING CONFLICTS IN FLUID

In a traditional WLAN that uses a fixed channel width, the conflict graph between N transmissions (all on the same channel) can be generated by performing pair-wise link throughput tests [71] at each PHY rate k, which requires a total of $O(N^2 \cdot k)$ measurements. Recent research [84, 133] has shown that this overhead can be reduced to $O(N \cdot k)$ using SINR based modeling. Applying such models to a variable channel width system is not straightforward, as the number of spectral overlaps (and hence interference) depends on the combinations of center frequencies and channel widths used. Figure 5.8 shows

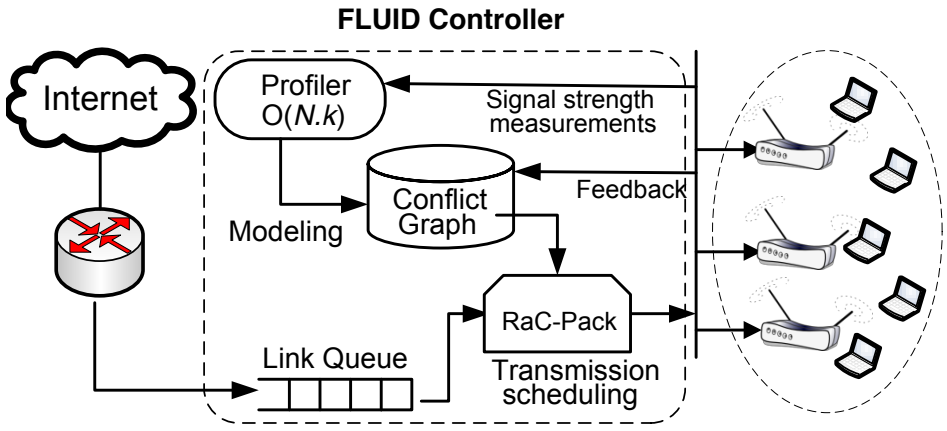

Figure 5.4: Flow of operations in FLUID. Periodic signal strength measurements are used to update the modeled conflict graph (§5.4). Packets arrive from the network gateway and are enqueued at a central controller. The controller releases these packets based on the transmission schedules derived by a packing algorithm (§5.5). APs receive the packets and transmit them according to the controller's prescribed flexible channel assignment, and subsequently notify the controller of all failures. The controller uses this feedback for scheduling retransmissions and refining the conflict graph.

two example spectrum overlap configurations. The number of distinct non-zero spectrum overlap configurations using the set of permissible center frequencies (as detailed in §5.3) for two links operating on channel widths w_1 and w_2 can be calculated as $(w_1 + w_2)/w_{\min} - 1$. Hence, the total number of spectrum overlap configurations taking into account $|w|$ possible widths are $\sum_{w_1} \sum_{w_2} \Big((w_1 + w_2)/w_{\min} - 1\Big)$, which evaluates to $2{\cdot}|w|{\cdot}(2^{|w|} - 1) - |w|^2$. Thus, computing the conflict graph using the approach in [71] would now require a significant overhead of $O(N^2{\cdot}k{\cdot}|w|{\cdot}2^{|w+1|})$, making it intractable for real systems. Next we show how our models significantly reduce this measurement overhead.

Modeling overview. The goal of the conflict graph module in Figure 5.4 is to predict the delivery ratio on a link (transmitter-receiver pair) in the presence of an interferer. It uses SINR based empirical models to predict the delivery

Modeling parameter	Definition
P_i	Transmitted power per unit Hz (at width w_i)
$\mathcal{A}(.)$	Signal attenuation function
$\widehat{S}_{tr}(w_i)$	Signal strength (in dBm) per hertz between transmitter t and receiver r using width w_i
$\Delta S(w_i, w_j)$	Modeled difference in received signal strength per hertz when switching from width w_i to width w_j
$\xi(w_i, w_j)$	Correction function applied to $\Delta S(w_i, w_j)$ for improving the model accuracy
$\mathcal{N}$	Noise floor per hertz
$\mathcal{I}_{t,r}(\tau, w_t, w_r)$	Quantifies the spectral overlap between a transmitter t using center frequency and width (f_t, w_t) and a receiver r using center frequency and width (f_r, w_r). Here $\tau = \lvert f_t - f_r \rvert$
$B_{r,w_r}(f)$	Band-pass filter's frequency response when a channel width of w_r MHz is used
$\widehat{\mathcal{D}}(.)$	Function used to predict delivery ratio based on modeled SINR

Table 5.3: Parameters used in FLUID's modeling procedure.

probabilities. In what follows, we first explain how our model computes the SINR for *perfect spectral overlap* case (the link and the interferer use the *same* center frequency and width), at all channel widths, using only measurements at a single width. We then extend the model to compute the SINR for *partial spectral overlap* case (the link and the interferer can use different center frequencies and widths). Finally, we derive the delivery prediction models using empirical measurements and use the computed SINR to model the delivery under interference. Figure 5.5 shows the overall modeling process. The parameters used in the modeling procedure are summarized in Table 5.3.

Interpolating SNR at different widths, using single width measurements. To compute the SINR at the receiver, we have to measure the signal strengths of the transmitter and the interferer at the receiver. However, as we show below, the received signal strength per hertz depends on the channel width. This would require us to carry out signal strength measurements at every channel width, resulting in a measurement overhead of $O(N \cdot |w|)$. We now show that it is possible to interpolate the received signal strength per hertz at different widths from measurements at only one width.

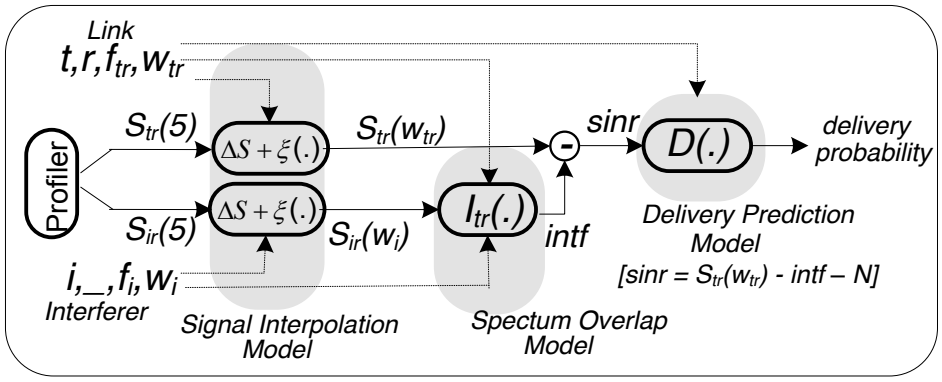

Figure 5.5: Sketch of the modeling process. Signal strengths of the transmitter and the interferer at their respective widths are interpolated using their corresponding signal strengths at 5 MHz. The amount of interference is then computed based on the spectral overlap, which is used to calculate the SINR. Finally, the SINR is input to the delivery prediction model to compute the delivery under interference.

Let P_i and P_j be the transmitted power per unit Hz at widths w_i and w_j respectively. Since the total power transmitted by the card is the same in both cases, we have $P_i \cdot w_i = P_j \cdot w_j$. Now, the signal strength per hertz at the receiver depends on the attenuation experienced by the wireless signal and is given by $s_i = \mathcal{A}(P_i)$. We can approximate the attenuation $\mathcal{A}(.)$ as $d^{-\alpha}P_i$, where α is the path-loss exponent [50]. We can compute the difference in received signal strength per hertz, $\Delta S(w_i, w_j)$ as $10\log(\frac{s_i}{s_j}) = 10\log(\frac{P_i}{P_j}) = 10\log(\frac{w_j}{w_i})$.

However, we observed that the difference in signal strength per hertz for our hardware only follow this relationship approximately. When we decreased the channel width from 40 MHz to 5 MHz, we observed $\Delta S(w_i, w_j)$ to be 8.6 dB on average, instead of 9 dB (per unit Hz). To account for this difference, we introduce a correction function $\xi(.)$. Let $\widehat{S}_{tr}(w_i)$ denote the signal strength per hertz (in dBm) between transmitter t and receiver r at width w_i, derived using empirical measurements. We have:

$$\widehat{S}_{tr}(w_i) = \widehat{S}_{tr}(w_j) + \Delta S(w_i, w_j) + \xi(w_i, w_j) \tag{5.1}$$

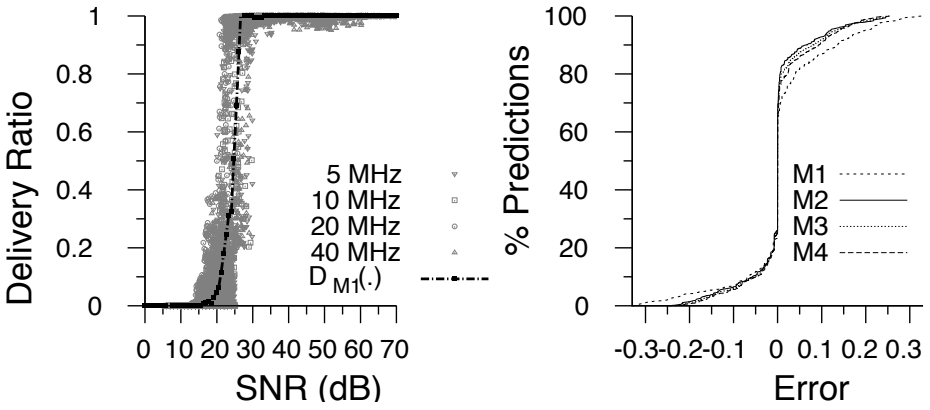

Figure 5.6: (left) Delivery ratio as a function of mean signal strength for different widths, across all the receivers at 6 Mbps. We show measured delivery ratio values and piece-wise linear interpolation as a function of SNR (model M1). (right) CDF of modeling error for all the four models.

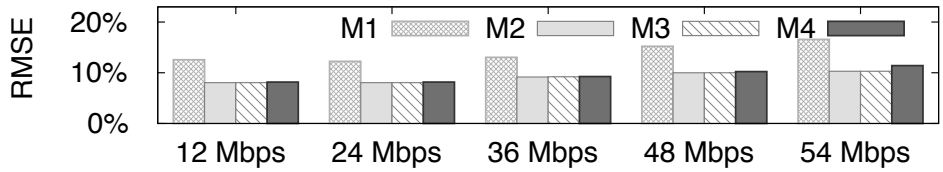

Figure 5.7: Prediction error for all the models at different PHY rates.

We empirically calculate the value of $\xi(.)$ using signal strength measurements from our testbed. We assume the noise floor per hertz ($\mathcal{N}$) to be constant and the signal to be evenly distributed over the transmitted bandwidth. We calculate the SNR at width w as $\widehat{\mathrm{S}}_{\mathrm{tr}}(w) - \mathcal{N}$. Figure 5.9 (left) shows the CDF of signal strengths at different widths for all links in our testbed. We observed that the difference between measured and theoretical signal strength per hertz values does not vary significantly, even for the most bursty link in our testbed. The observed mean/std. deviation values *across all links* for $\xi(40,5)$ were $-0.34/0.13$ dB, that for $\xi(20,5)$ were $-0.13/0.12$ dB, and finally for $\xi(10,5)$ were $-0.08/0.16$

dB (per unit Hz). Since these variations are low, in our model, we account for the difference in the measured and theoretical signal strength per hertz using the mean value of $\xi(.)$. We note that this calibration is a one-time overhead that is necessitated when using the model for the first time with a particular hardware chipset. Once the mean values of $\xi(.)$ are determined, these can be used to accurately and efficiently estimate SNR as follows. Instead of carrying out the signal measurements at every width, we carry out $O(N)$ signal measurements at the lowest width of 5 MHz (as it has the longest range), and use Equation 5.1 to derive the SNR at all other widths.

Modeling SINR for perfect spectral overlaps at all widths. To model SINR in the presence of an interferer, using width w, we first interpolate the signal strength per hertz of the transmitter to the receiver, and that of the interferer to the receiver i.e., we use Equation 5.1 to interpolate $\widehat{S}_{tr}(w)$ and $\widehat{S}_{ir}(w)$ from corresponding signal measurements at 5 MHz, $\widehat{S}_{tr}(5)$ and $\widehat{S}_{ir}(5)$. Now, the SINR can simply be calculated as $\widehat{S}_{tr}(w) - \widehat{S}_{ir}(w) - \mathcal{N}$ dB. We now provide extensions to the previous model, to quantify the amount of interference for the partial overlap case where the links can use any permissible center frequencies and channel widths.

Modeling SINR for partial spectral overlaps at all widths and frequencies. To characterize the amount of interference experienced by a receiver r using a width w_r and a center frequency f_r, from an interferer t using a width w_t and center frequency f_t, we extend the model developed in [106] to calculate the *interference factor*, $\mathcal{I}_{t,r}(.)$ for a variable channel width system. $\mathcal{I}_{t,r}(.)$ quantitatively captures the amount of spectral overlap between the interferer and the receiver by calculating the area of intersection between a signal's spectrum and a receiver's band-pass filter. We incorporate the interferer and receiver channel bandwidths, w_t and w_r into this model to derive $\mathcal{I}_{t,r}(.)$:

$$\mathcal{I}_{t,r}(\tau, w_t, w_r) = \int_{-\infty}^{+\infty} T_{t,w_t}(f) B_{r,w_r}(f - \tau)\, df \tag{5.2}$$

In above equation, the parameter τ represents the difference in the center frequencies of the channels i.e., $\tau = f_t - f_r$. The parameter $T_{t,w_t}(f)$ denotes the transmitted signal's power distribution across the frequency spectrum when a channel bandwidth of w_t MHz is used. We approximate $T_{t,w_t}(f)$ with the corresponding transmit spectrum mask [106]. Finally, $B_{r,w_r}(f)$ denotes the band-pass filter's frequency response when a channel of w_r MHz is used. Assuming the receive filter for a particular bandwidth to be same as the transmit spectrum mask [106], for 802.11a we get:

$$B_{r,w_r}(f) = T_{t,w_t}(f) =$$

$$\begin{cases} -40\text{dB} & \text{if } |f - F_c| \geqslant (30/\mathcal{B})\text{MHz} \\ -28\text{dB} & \text{if } (20/\mathcal{B})\text{MHz} \leqslant |f - F_c| < (30/\mathcal{B})\text{MHz} \\ -20\text{dB} & \text{if } (11/\mathcal{B})\text{MHz} \leqslant |f - F_c| < (20/\mathcal{B})\text{MHz} \\ 0\text{dB} & \text{otherwise} \end{cases} \tag{5.3}$$

where F_c denotes the channel center frequency and bw is the channel bandwidth (w_t or w_r) used and $\mathcal{B}$ is the bandwidth scaling factor calculated as $\mathcal{B}$=20/bw.

Now for two links (t_1, r_1) and (t_2, r_2) using center frequencies and widths (f_1, w_1) and (f_2, w_2), the amount of interference experienced by r_1 can be characterized as $\mathtt{intf} = \widehat{S}_{t_2 r_1}(w_2) + 10\log(\mathcal{I}_{t_2,r_1}(|f_2 - f_1|, w_2, w_1))$ dB. The effective SINR would be $\widehat{S}_{t_1 r_1}(w_1)$-$\mathtt{intf}$-$\mathcal{N}$ dB.

Predicting delivery ratio. In the last step of our modeling process, we predict the delivery ratio for a link using the SINR estimated earlier. We first show the relationship between SNR and the delivery ratio for an isolated link when using different widths, and then derive delivery prediction models.

Delivery under isolation. We perform $O(N \cdot |w| \cdot k)$ measurements where each node broadcasts in turn at all widths and rate combinations, and the remaining nodes measure the average signal strengths and corresponding delivery ratios. All nodes use the same center frequency and channel width. Figure 5.6 (left) shows the SNR vs. delivery ratio for 231 link pairs for each of the four channel

widths at 6 Mbps.[3] For values of SNR greater than 26 dB, the delivery ratio is close to 1, whereas for SNR less than 18 dB, the deliver ratio is close to 0; for intermediate values of SNR, the delivery ratio increases with signal strength. This behavior is *similar* across widths, since for a given signal strength, the probability that a packet is successfully decoded is independent of width. Furthermore, we observed a stronger correlation between SNR and delivery ratio when viewed across individual receivers.

Based on this, the most relevant parameters for modeling delivery are: SNR, channel width and the receiver under consideration. In light of this, we explored four models to derive the delivery prediction function $\widehat{\mathcal{D}}(.)$. In M1, we model the delivery ratio as a piece-wise linear function of SNR. In M2, we used receiver-specific curves including the SNR and channel width. M3 only used receiver-specific curves along with SNR. M4 is similar to M3, except that SNR is computed using Equation 5.1.

Delivery under interference. To predict the delivery under interference, we compute the SINR using the techniques mentioned before and feed this into one of the four delivery prediction models. We now evaluate the accuracy of these models in the presence of an interferer for the perfect spectral overlap case.

In order to measure the ground truth, we carry out the following $O(N^2 \cdot k \cdot |w|)$ measurements: we pick a pair of nodes in turn, and both of them simultaneously transmit data while the rest of the nodes measure the signal strengths and corresponding delivery ratios. This process is repeated for all channel width and rate combinations. We note that all nodes use the same center frequencies and widths. Figure 5.6 (right) shows the CDF of the error for all the four models at 6 Mbps, and Figure 5.7 shows the RMSE (root mean square error) for the models across different PHY rates. We observe that all the four models perform reasonably well. Models M2, M3, and M4 have lower error compared to M1, owing to the use of receiver specific curves. For these models, the error is less than 10% for 90% of the predictions, with maximum error being less than 30% (Fig. 5.6 (right)). The overall RMSE for all the models were: 14.2%, 8.7%,

[3]This behavior also holds for all the other rates. The SNR curves are shifted to the right, as higher rates require a higher SNR to decode a packet correctly.

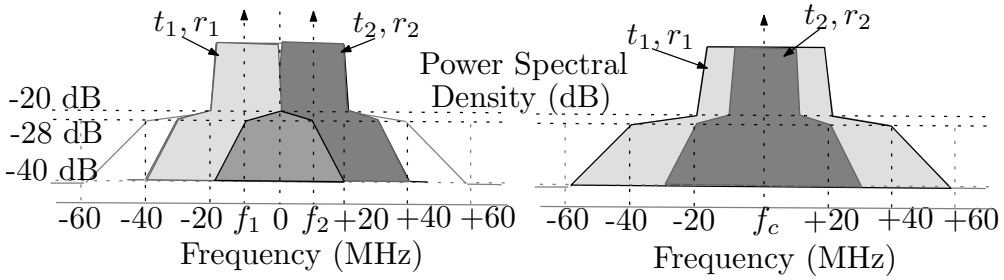

Figure 5.8: Example spectrum overlap scenarios. (left) Two links (t_1, r_1) and (t_2, r_2) using a channel width of 20 MHz and center frequencies f_1 and f_2 separated by 20 MHz. (right) Two links (t_1, r_1) and (t_2, r_2) using the same center frequency (f_c), but different channel widths of 40 MHz and 20 MHz respectively.

8.9%, and 9.6%. We observe that M2 and M3 have very similar performance, confirming that the delivery ratios were independent of the width used. More importantly, M4 which uses signal interpolation has an accuracy which is quite close to M2. This is a useful result as it helps us reduce the conflict graph computation overhead to $O(N \cdot k)$ for a network where all links can operate on any width while using the same center frequency. We therefore choose M2 for delivery prediction in FLUID. We also evaluated the models for the partial overlap case, and observed similar delivery prediction accuracy numbers.

Packing accuracy. We now evaluate both the partial and perfect spectrum overlap cases using a more intuitive measure — error in predicting the minimum frequency separation required to resolve the conflict between any two links. Note that over-predicting the frequency separation leads to poor usage of spectrum, while under-prediction can result in throughput degradation.

We experimented with 500 link-interferer (tr-i) combinations (across different PHY rates) in our testbed, where the link and the interferer can use any widths, w_{tr} and w_i. In each case we measured $\widehat{f}_{min}$, the minimum frequency separation required between the link and the interferer such that the conflict is resolved. We also compute the predicted separation f_{min} using the $\mathcal{I}_{t,r}(.)$ model, and a naive packing approach where the center frequencies are simply separated by $(w_{tr} + w_i)/2$ MHz. We then compute the difference

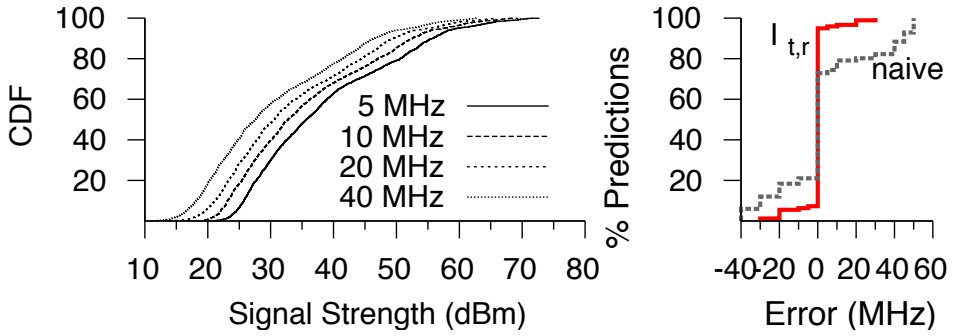

Figure 5.9: (left) CDF of signal strengths in the testbed for different channel widths. (right) CDF of error in estimating the minimum channel separation (Δf_{min}) across different PHY rates and channel width combinations, using naive and $\mathcal{I}_{t,r}$ models.

in the measured and predicted frequency separation $\Delta f_{min} = f_{min} - \widehat{f}_{min}$. Figure 5.9 (right) shows the CDF of Δf_{min} for both the models. The $\mathcal{I}_{t,r}(.)$ model results in better spectrum reuse by predicting Δf_{min} correctly in 87.6% of the cases. The naive model predicts only 52% of the cases accurately.

Example scenarios. We now present two example scenarios and show how $\mathcal{I}_{t,r}(.)$ can be used to model the interference.

Example 1 - How much frequency separation is needed? Consider the packing in Figure 5.8 (left), where the two 20 MHz links (t_1, r_1) and (t_2, r_2) have their center frequencies separated by of 20 MHz. Is the frequency separation enough? To answer this we have to estimate the delivery ratio at both the receivers for this packing: the interference factor for r_1 in this case turns out to be $\mathcal{I}_{t_2,r_1}(20, 20, 20)$ $= 0.0571$, which is reduction in the interferer signal by 12.43 dB. Hence, the SINR at r_1 in this case would be $\widehat{S}_{t_1 r_1}(20)$-$\widehat{S}_{t_2 r_1}(20)$+12.43 dB. If this is not sufficient for $\widehat{\mathcal{D}}(r_1, \text{SINR})$ to be close to 1, then the links have be separated further apart. For e.g., at 25 MHz, the $\mathcal{I}_{t_2,r_1}(25, 20, 20) = 0.0081$ which results in SINR at r_1 of $\widehat{S}_{t_1 r_1}(20) - \widehat{S}_{t_2 r_1}(20) + 20.91$ dB i.e., an increase in SINR by around 8.5 dB which might sufficient for the delivery ratio to be close to 1. Similarly, we can estimate the frequency separation needed for r_2.

Example 2 - Can we improve spatial reuse by narrowing widths? Consider two 40 MHz links (t_1, r_1) and (t_2, r_2) which are on the same center frequency. The interference factor for both receivers is 1, SINR at r_1 is $\widehat{S}_{t_1 r_1}(40)$-$\widehat{S}_{t_2 r_1}(40)$ and that at r_2 is $\widehat{S}_{t_2 r_2}(40)$-$\widehat{S}_{t_1 r_2}(40)$. How does the interference relationship change when (t_2, r_2) switches to 20 MHz, while not changing the center frequency? The resulting packing is shown in Figure 5.8 (right). When (t_2, r_2) switches to 20 MHz, the interference factor for r_2 remains the same (i.e., $\mathcal{I}_{t1,r2}(0, 40, 20) = 1$), since there is a complete spectral overlap for r_2. However, $\widehat{S}_{t_2 r_2}(20)$ is around 3 dB higher than $\widehat{S}_{t_2 r_2}(40)$ (Equation 5.1). This results in SINR at r_2 increasing by 3 dB. Whereas, for r_1 , the interference factor $\mathcal{I}_{t2,r1}(0, 20, 40) = 0.49$, which is around -3.1 dB. Therefore, the new SINR becomes $\widehat{S}_{t_1 r_1}(40) - \widehat{S}_{t_2 r_1}(20) -$ 3.1 dB, i.e., $\widehat{S}_{t_1 r_1}(40) - \widehat{S}_{t_2 r_1}(40) + 3 - 3.1$ dB implying, that the SINR at r_1 almost remains the same. That is, by narrowing the channel width of a link, we could improve its SINR without affecting the SINR of the other link. In our experiments, we found a number of such instances where narrowing channel width can provide opportunities for spatial reuse (§5.7).

External interference. In order to handle external interference, the model can be modified as follows: the receivers can measure the increased noise floor, and report this back along with the signal strength measurements. The SINR computation can then use a receiver specific noise floor $\mathcal{N}_r$, instead of a constant $\mathcal{N}$ to improve the accuracy of delivery prediction.

Summary. We sketch the modeling process in Figure 5.5. We carry out $O(N \cdot k)$ measurements at the lowest channel width, 5 MHz. In order to predict the delivery ratio of link in the presence of an interferer, we first interpolate the signal strengths of the transmitter and the interferer at their widths. Based on the spectral overlaps, we compute the interference using the $\mathcal{I}_{t,r}(.)$ model. Finally, we calculate the SINR, which is then input to $\widehat{\mathcal{D}}(.)$ to estimate the delivery probability.

Assume that a set of packets arrive at the FLUID controller. Now, based on the conflict graph, the next step for the controller is to "pack" the transmissions i.e., determine the subset of packets that can be scheduled for transmission simultaneously, along with an assignment of the center frequencies and channel widths. In FLUID, such a decision is made at the time granularity of an *epoch*. We discuss the factors that determine the epoch duration in §5.6.

Scheduling complexity: The scheduling problem to optimize throughput by assigning appropriate time-frequency blocks is NP-hard [168]. The size of this problem is $\sum_{r=1}^{r=N} \binom{N}{r} |\mathcal{F}|^r$, where $\binom{N}{r}$ is number of the ways in which the controller can pick r out of N transmissions, and $|\mathcal{F}|^r$ is all possible frequency and width combinations for r APs.

Packing heuristics. In order to reduce the search space in scheduling, we use two heuristics explained below:

Throughput estimation: The throughput estimation algorithm, *estimateTput()* (Algorithm 1) takes a set of packed transmissions $\mathcal{T}=(t, r, f, w)$, and returns a vector of estimated individual transmission throughputs. The throughput of an individual transmission T_i is calculated as follows: the effective signal strength from each of the other $\mathcal{T}$-$\{T_i\}$ transmissions is calculated using the modeling techniques presented in §5.4, and is summed up to calculate the total interference (lines 8-10). This then used to compute the SINR. Finally, the controller uses the SINR to estimate the throughput by picking the best PHY data rate (lines 14-19): it iterates through the delivery ratio curves for each data rate, and picks the rate which maximizes the throughput (data rate $\times$ delivery probability).

RaC-Pack: In FLUID, the central controller uses a randomized algorithm, RaC-Pack (Randomized Compaction based Packing) to derive the transmission schedules. RaC-Pack (Algorithm 1) takes the FIFO queue of packets at the controller as input and creates a set of packed transmissions for each epoch. We first describe the compaction step that can be applied to a packed transmission set so as to maximize a particular objective.

Algorithm 1: Model based Throughput Estimation (estimateTput)

Input : Set of packed transmissions $\mathcal{T} = \{t, r, f, w\}$, Delivery ratio model, $\widehat{\mathcal{D}}(.)$, Signal correction model, $\xi(.)$
Output: Estimated throughput for $\mathcal{T}$

1 $\overrightarrow{tv}_{tot} \leftarrow 0$
2 **foreach** $T_i = (t_i, r_i, f_i, w_i) \in \mathcal{T}$ **do**
3 $\quad (d_{T_i}^{\mathcal{T}-\{T_i\}}, dr_{best}) \leftarrow \rho(T_i, \mathcal{T} - \{T_i\})$
4 $\quad \overrightarrow{tv}_{tot}[i] \leftarrow (d_{T_i}^{\mathcal{T}-\{T_i\}} * dr_{best})$
5 **return** $\overrightarrow{tv}_{tot}$
6 **Procedure** $\rho(T_i, \mathcal{T}')$**:**
7 $\quad$ Let $T_i = \{t_i, r_i, f_i, w_i\}$; $intf \leftarrow 0$
8 $\quad$ **foreach** $T_j = (t_j, r_j, f_j, w_j) \in \mathcal{T}'$ **do**
9 $\quad\quad S_{t_j,r_i}(w_j) \leftarrow \widehat{S}_{t_j,r_i}(5) + \Delta S(w_j, 5) + \xi(w_j, 5)$
10 $\quad\quad S'_{t_j,r_i} = S_{t_j,r_i}(w_j) + 10\log(\mathcal{I}_f(|f_j - f_i|, w_j, w_i))$ $intf = intf + S'_{t_j,r_i}$
11 $\quad S_{t_i,r_i}(w_i) \leftarrow \widehat{S}_{t_i,r_i}(5) + \Delta S(w_i, 5) + \xi(w_i, 5)$
12 $\quad sinr = S_{t_j,r_i}(w_i) - intf$
13 $\quad max\_tput \leftarrow 0$
$\quad$ `/* SINR based rate adaptation */`
14 $\quad$ **foreach** $dr \in R$ **do**
15 $\quad\quad cur\_tput \leftarrow dr * \widehat{\mathcal{D}}(r_i, dr, sinr)$
16 $\quad\quad$ **if** $cur\_tput > max\_tput$ **then**
17 $\quad\quad\quad max\_tput \leftarrow cur\_tput$
18 $\quad\quad\quad dr_{best} \leftarrow dr$
19 $\quad\quad\quad d_T^{\mathcal{T}'} \leftarrow \widehat{\mathcal{D}}(r_i, dr, sinr)$
20 $\quad$ **return** $(d_T^{\mathcal{T}'}, dr_{best})$

Compaction Step: Keeping the center frequency and width assignments of all the other transmissions the same, the compaction step (lines 22-29) assigns a center frequency and width to a particular transmission, T_i that maximizes a criteria (lines 24-28). We supply the objective function (*computeOBJ*) with one of the following two criteria: (i) maximize the total throughput (FLUID-thr) or (ii) find the best min-max throughput (FLUID-fair) which results in better fairness, at the cost of throughput. The function *estimateTput*, is used to estimate throughput during each iteration (line 26).

Algorithm 2: RaC-Pack: Transmission Packing

Input : fifoQ (FIFO queue of packets), $vQ_1 \dots vQ_n$ (per-client virtual packet queues), $\mathcal{F} = \{(f, w)\}$ (set of frequency f, width w combinations)
Output : Set of packed transmissions $\mathcal{T}_{next} = \{(t, r, f, w)\}$

$\mathcal{T}_{next} \leftarrow 0, \mathcal{T}_{cur} \leftarrow 0$
$p_{head} \leftarrow$Dequeue(fifoQ); $(f_1, w_1) \leftarrow \mathcal{F}[0]$
$T_1 \leftarrow (tx(p_{head}), rx(p_{head}), f_1, w_1)$; $packedAPs \leftarrow tx(p_{head})$
$(\mathcal{T}_{next}, \overrightarrow{tv}_{best}) \leftarrow$ `COMPACTION`$(\{T_1\}, 0)$; $\mathcal{T}_{cur} \leftarrow \mathcal{T}_{next}$
$r_i \leftarrow$ `RAND`$(0 \dots n-1)$
for i *in* $0 \dots n$ **do**
  $next \leftarrow (r_i + i) \bmod n$
  $p_{next} \leftarrow$Dequeue(vQ_{next})
  if $tx(p_{next}) \in packedAPs$ **then**
    continue
  $T_{next} \leftarrow (tx(p_{next}), rx(p_{next}), f_1, w_1)$
  $\mathcal{T}_{cur} \leftarrow \mathcal{T}_{next} \bigcup T_{next}$
  while $\mathcal{T}_{cur} \neq \mathcal{T}_{prev}$ **do**
    $\mathcal{T}_{prev} \leftarrow \mathcal{T}_{cur}$; $k \leftarrow |\mathcal{T}_{cur}|$
    $r_j \leftarrow$ `RAND`$(0 \dots k-1)$
    for j *in* $0 \dots k$ **do**
      $next' \leftarrow (r_j + j) \bmod k$
      $(\mathcal{T}_{cur}, \overrightarrow{tv}_{cur}) \leftarrow$ `COMPACTION`$(\mathcal{T}_{cur}, next')$
  if `computeOBJ` $(\overrightarrow{tv}_{cur}, \overrightarrow{tv}_{best}$, criteria) **then**
    // $\overrightarrow{tv}_{cur}$ improves over $\overrightarrow{tv}_{best}$ for a given criteria
    $\overrightarrow{tv}_{best} \leftarrow \overrightarrow{tv}_{cur}$; $\mathcal{T}_{next} \leftarrow \mathcal{T}_{cur}$; $packedAPs \leftarrow packedAPs \bigcup tx(p_{next})$
return $\mathcal{T}_{next}$;
Procedure `COMPACTION` $(\mathcal{T}, i)$**:**
  $\overrightarrow{tv}_{bestlocal} \leftarrow 0$; $\mathcal{T}' \leftarrow \mathcal{T}$
  foreach $(f, w) \in \mathcal{F}$ **do**
    $\mathcal{T}[i] \leftarrow (t_i, r_i, f, w)$
    $\overrightarrow{tv}_{cur} \leftarrow$ `estimateTput`$(\mathcal{T})$
    if `computeOBJ` $(\overrightarrow{tv}_{cur}, \overrightarrow{tv}_{best}$, criteria) **then**
      $\overrightarrow{tv}_{bestlocal} \leftarrow \overrightarrow{tv}_{cur}$; $\mathcal{T}' \leftarrow \mathcal{T}$
  return $(\mathcal{T}', \overrightarrow{tv}_{bestlocal})$;

The RaC-Pack scheduling algorithm works as follows: In order to prevent starvation, RaC-Pack always schedules the first packet in FIFO queue for transmission in the current epoch. It then applies the compaction step to this transmission to find the 'best' packing (lines 2-4). Next, the algorithm goes through the rest of the transmissions in a randomized order, and adds them to the transmission schedule if they improve the throughput (lines 5-20). This is done by adding a transmission to the currently packed set, and then repeatedly invoking the compaction step for the each of the transmissions in succession. The order of invocation is randomized by using a random permutation of the transmissions. This compaction process (lines 13-18) is repeated until the objective function stops improving. We note that this iterative process will converge, as in each iteration, the objective function progressively improves the throughput vector based on the specified criteria. The total number of rounds for the algorithm can vary with the topology and traffic pattern, and the worst case complexity is $O(|\mathcal{F}|^N)$. We set an upper bound of 50 rounds, and in our experiments with different topologies, we found that the algorithm converges after approximately 21.3 rounds on an average. In §5.7, we compare RaC-Pack to the brute-force approach of evaluating all possible schedules.

5.6 IMPLEMENTATION ASPECTS

Our implementation of FLUID consists of: (a) a central controller that generates the conflict graph and uses the RaC-Pack algorithm to schedule packets. We have implemented this on a Linux PC (3.33 GHz dual core Pentium IV, 2 GB DRAM) (about 3500 lines of C code and a few hundred lines of Perl scripts). (b) Soekris based wireless APs and clients, modified to implement channel and width switching functionality. The scheduler is a kernel module that utilizes *high-resolution* timers. In order to reduce communication path latencies, we have implemented a direct path between the Ethernet and WiFi drivers for the APs. This allows packets received on the wired interface to be immediately forwarded to the wireless interface, bypassing the kernel network queue. We also made driver modifications to ensure that transmit buffers are not flushed, and that clients do not disassociate with the AP when switching frequencies

or widths. We now highlight some of the other implementation aspects and system design issues that arise when deploying FLUID.

Handling Uplink Transmissions. To account for uplink (client-to-AP) transmissions, we use a two-phase TDMA approach [103, 105]: the first phase uses flexible channelization for downlink traffic, and the second phase is for uplink traffic using DCF . The controller *adapts* the time for each phase according to the downlink/uplink traffic ratio (based on queue lengths). By default, since most traffic in enterprise WLANs is downlink [148], we use a 4:1 ratio between the downlink and uplink phases. Carrier sensing and ACKs are disabled in the downlink phase, since they add overheads in a TDMA MAC [103]. Instead, we use block ACKs that are transmitted in the uplink phase. FLUID controller uses this feedback to schedule retransmissions and to refine the modeled conflict graph. To assign channel widths and frequencies in the uplink phase, we use a simple approach: each AP groups its clients into one of four channel widths, based on the widest channel width each client can successfully communicate on. During the uplink phase, FLUID APs switch to their respective center frequencies, and operate on one of the channel widths; over time, the APs cycle through all channel widths with average dwell times at each width being proportional to aggregate uplink traffic from each group. We realize that an optimal assignment for the uplink phase is a challenging problem, and are actively investigating solutions to this problem.

Association. APs are modified to beacon at the lowest channel width of 5 MHz, which has the most range. The center frequencies for beacon transmissions are decided using RaC [107], a conflict-aware fixed-width channel assignment mechanism. Client drivers are modified to perform passive scans using a width of 5 MHz. In our current implementation, we do not support active scanning.

Co-ordinated switching. To inform the clients about their future schedules, APs use the 802.11 Beacon Information Element (BIE). BIE consists of a list of [*e*poff, phase, chan, clist] where *e*poff is the epoch offset, *p*hase indicates uplink or downlink, *c*han is the frequency and width, and *c*list is the list of clients for which traffic has been scheduled in the epoch. To account for beacon losses, the APs also insert a layer 2.5 header in the data packets with information about

future schedules. We use built-in Atheros clock synchronization to synchronize the epoch boundaries at APs and the clients.

Implementation overheads. We instrumented the drivers to calculate the delays in controller-AP-client communication path and channel/width switching. We observed that the overheads are dominated by the channel and width switching component; the mean/std. deviation for which was 4.11/0.244 ms. To amortize these overheads, (i) we set the epoch duration to 6 ms, and (ii) we use two interfaces at the APs. While one interface is active during an epoch (i.e., it is involved in communication), the other interface prepares for the next epoch. These switching overheads could reduce in future; emerging wireless cards have switching latencies of less than 100 μs [6], while prior work in solid state electronics has shown that this delay can be reduced to as low as 40 μs [46]. Finally, in order to maintain an accurate conflict graph that can take into account the dynamics of the environment, it is important that the signal strengths are frequently updated. Since there is little external interference in our experimental testbed, which is also likely in other enterprise networks, we chose a measurement periodicity of 10 seconds. However, this is a tunable parameter, and in a more noisy environment one could reduce the measurement periodicity. Similar to previous systems like DIRC [103] and CENTAUR [148], each measurement instance in FLUID lasts for 4 ms. We note that the results in §5.7 include these measurement overheads.

5.7 EVALUATION

Our testbed evaluation aims at characterizing the throughput improvements with FLUID and demonstrate its feasibility on commodity 802.11 hardware. We first evaluate FLUID over a large number of canonical topologies to systematically characterize the performance gains that stem from different components. We show the results for both max-throughput (FLUID-thr) and best min-max throughput (FLUID-fair). Next, we evaluate FLUID over a 23 node representative topology and quantify the performance gains. We perform the experiments at different fixed PHY rates and with dynamic rate adaptation. When using rate adaptation, we run DCF and CENTAUR using SampleRate,

Section	Evaluation component and set up	Summary of results
Microbenchmarking the model		
§ 5.4 (Fig. 5.7, Fig. 5.6(b))	Accuracy of Delivery Models (231 link pairs, different PHY rates)	RMSE: 14.2%, 8.7%, 8.9%, and 9.6%
§ 5.4 (Fig. 5.9)	Packing Accuracy (500 link-intf comb., different PHY rates)	Accuracy: 87.6% ($\mathfrak{I}_{t,r}$), 52% (naive)
§ 5.7 (Fig. 5.11)	Gains w/ packing (331 2-link topologies, Rate: Auto/different PHY rates)	Median gain (Thr./MHz): 1.51× (best DCF)
Gains in specific scenarios		
§ 5.7 (Fig. 5.10(a), Tab. 5.5)	Clients with differing SNRs (241 1 AP-2 client topologies, Rates: Auto/different PHY rates)	Up to 1.68×, Median: 1.40× (best DCF), 1.38× (CENTAUR)
§ 5.7 (Fig. 5.10(b), Tab. 5.6)	Clients under interference (194 2-link topologies, Rates: Auto/different PHY rates)	Up to 2×, Median: 1.35× (best DCF), 1.32× (CENTAUR)
§ 5.7 (Fig. 5.12(a))	Hidden links (unscheduled Vs. scheduled) (346 2-link topologies, Rates: Auto/different PHY rates)	Median: 1.74× (best DCF), Avg: 1.88× (DCF-flex), up to 1.47× (CENTAUR)
§ 5.7 (Fig. 5.12(b))	Exposed links (unscheduled Vs. scheduled) (346 2-link topologies, Rates: Auto/different PHY rates)	Up to 1.97× (best DCF), Median: 1.51× (best DCF), 1.51× (DCF-flex), 1.31×(CENTAUR)
Gains on a representative topology		
§ 5.7 (Fig. 5.13, Tab. 5.7)	UDP throughput (23-node topology, Rate:Auto/different PHY rates)	Median: 1.59× (best DCF), 1.34× (CENTAUR)
§ 5.7 (Fig. 5.13)	TCP throughput (23-node topology, Rate:Auto)	Mean: 1.63×(best DCF), 1.33×(CENTAUR)
§ 5.7	Performance of Rac-Pack (23-node topology, Rate:Auto)	0.95× (brute force), 0.98× (Rac-Pack w/ actual CG)

Table 5.4: Summary of the results. Gain is reported for throughput unless otherwise noted.

Scheme	Gains over best DCF config. 12 Mbps	36 Mbps	54 Mbps	Gains over CENTAUR 12 Mbps	36 Mbps	54 Mbps
FLUID-thr	1.43×	1.39×	1.47×	1.41×	1.42×	1.46×
FLUID-fair	1.28×	1.21×	1.26×	1.23×	1.27×	1.25×

Table 5.5: Median gains (from using client-centric widths) over best DCF configuration and CENTAUR.

and for FLUID, we use the SINR based rate adaptation mechanism (§5.5). We assume that a total of 40 MHz spectrum is available. We quantify the gains of FLUID over DCF with fixed channel width configurations i.e., (i) DCF using a single 20 or 40 MHz channel (DCF-20 or DCF-40) and (ii) DCF using two 20 MHz channels and RaC-based channel assignment [107], denoted by DCF-2x20. To understand the gains attributable to flexible channelization (i.e., variable channel widths and packing) alone, we also compare with DCF employing flexible channelization (DCF-flex) and CENTAUR, a fixed channel width centralized scheduling (TDMA) approach which can exploit exposed terminals [148]. In our experiments, we operate CENTAUR at 40 MHz. The traffic on all the links is backlogged. We report the aggregate throughput in each case, and use Jain's Fairness Index [73] to report overall fairness. Table 5.4 summarizes the results presented in the chapter.

Gains from using client-centric widths

FLUID improves the throughput by using client-centric, link quality width aware assignment (e.g., case E1 in §5.2). To evaluate the gains from this aspect, we experiment with 241 single AP-two client topologies with both the clients having SNRs that differ by at least 3 dB. When experimenting with different rates, we only considered cases where the delivery probability of both links was greater than 0.9 at 20 MHz.

— *Different PHY rates:* Table 5.5 shows that FLUID-thr and FLUID-fair achieve median throughput gains of 44% and 26% over the best DCF configuration (DCF-20 or DCF-40), and 41% and 27% over CENTAUR across different PHY rates. CENTAUR and DCF-40 do not perform well, as the throughput of the lower SNR client suffers when using a 40 MHz channel. Although DCF-20

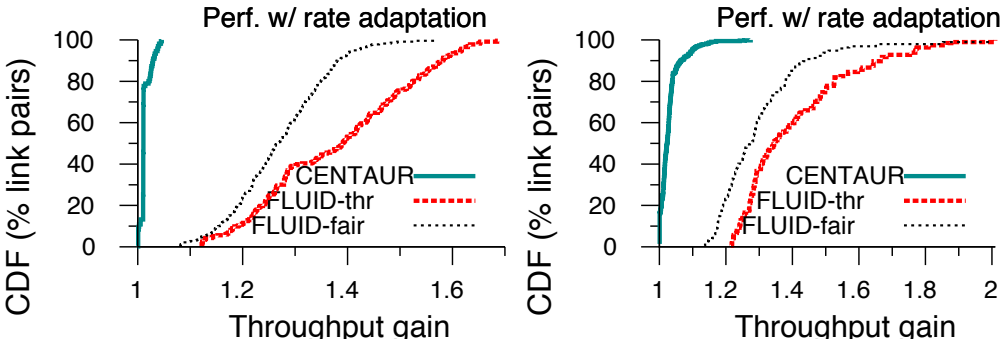

Figure 5.10: Throughput gains with rate adaptation for CENTAUR, FLUID-thr and FLUID-fair over DCF with fixed channel widths from (left) link quality aware width assignment (241 single AP - two client topologies) (right) increased transmission concurrency (194 two-link topologies with varying degrees of conflict).

improves the SNR by operating the links at 20 MHz, the overall throughput reduces due to spectrum wastage. FLUID operates the higher SNR link at 40 MHz, and the lower SNR link at 20 MHz, with the AP switching between these two widths. FLUID-fair provides lesser gains in order to improve fairness. Fairness indices [73] for FLUID-thr and FLUID-fair were 0.9 and 0.99, while those for DCF-40 and CENTAUR were 0.56 and 0.99.

— *Rate adaptation:* Figure 5.10 (left) shows the CDF of throughput gains for CENTAUR and FLUID over the best DCF configuration. We observe that FLUID-thr and FLUID-fair can improve the aggregate throughput up to 68% and 55% over DCF respectively, while improving fairness. The median gains over DCF were around 40% and 25%, while those over CENTAUR were 38% and 23%. The individual throughput gains for the lower SNR link in these cases were much higher.

Scheme	Gains over best DCF config. 12 Mbps	36 Mbps	54 Mbps	Gains over CENTAUR 12 Mbps	36 Mbps	54 Mbps
FLUID-thr	1.41×	1.47×	1.49×	1.32×	1.39×	1.46×
FLUID-fair	1.23×	1.29×	1.26×	1.2×	1.26×	1.27×

Table 5.6: Median gains under interference across different PHY rates for FLUID-thr and FLUID-fair over the best DCF configuration and CENTAUR for 194 topologies.

Gains under interference

FLUID improves the network throughput by choosing widths which result in increased transmission concurrency under interference (e.g., case E3 in §5.2). To illustrate this, we experiment with 194 one-way hidden interference cases.

— *Different PHY rates:* Here, DCF-40 is unable to resolve the conflict and performs poorly. DCF-2x20 resolves it by assigning the links different channels, whereas CENTAUR does so by serializing the transmissions. However, in many cases, *narrowing the channel width resolves the conflict* due to increase in SINR. FLUID-thr always operates the interfering link at 40 MHz and the other link at 20 MHz, thus allowing *simultaneous* transmissions. To achieve better fairness, FLUID-fair periodically reserves an epoch for the interfered link. Table 5.6 shows that FLUID-thr and FLUID-fair achieve consistent gains (up to 49% and 29%) across different PHY rates due to increased transmission concurrency.

— *Rate adaptation:* Figure 5.10 (right) shows that with rate adaptation, FLUID provided up to 2× gains over the best DCF configuration. Median gains for FLUID-fair and FLUID-thr were 28% and 35%. Corresponding gains over CENTAUR were 26.2% and 32%. In some cases, CENTAUR performs better than DCF-2x20 as operating the links on two adjacent 20 MHz channels was not enough to reduce the conflict.

Gains from conflict-aware packing

FLUID's gains also stem from (i) efficient transmission packing using partial spectral overlaps and (ii) avoiding harmful packing by separating the center

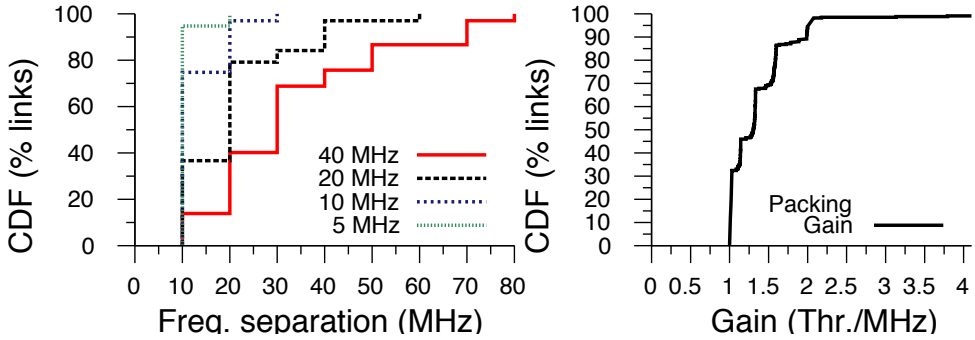

Figure 5.11: (left) Minimum center frequency separation required for conflict resolution at different widths (right) CDF of gain from intelligent packing across 331 link pairs.

frequencies by at least f_{min} (§5.4). We experimented with 331 two-link topologies, where both the links were using the same channel width w. Figure 5.11 (right) shows that f_{min} for these links varies — some links benefit from efficient packing ($f_{min} < w$) and others need greater frequency separation ($f_{min} > w$). Figure 5.11 (right) shows the CDF of packing gain in terms of throughput per unit MHz. For links with $f_{min} > w$, a maximum gain of $4\times$ (with median 51%) was observed as DCF suffered from losses due to interference. For links with $f_{min} < w$, efficient packing resulted in gains up to 70%.

Unscheduled and scheduled approaches

To compare scheduled and unscheduled approaches employing flexible channelization, we experiment on 346 two-link topologies that fall into one of two categories (when using 40 MHz): (a) conflicting links or hidden terminals (b) non-conflicting links and exposed terminals. We compare FLUID with three schemes: (1) DCF-fixed: this is the best amongst all DCF configurations where both the links use the same channel width (2) DCF-flex: this is the best amongst all DCF configurations where links can use any permissible combination of

channel widths and frequencies, and (3) CENTAUR operating on single channel of 40 MHz.

— *Conflicting/Hidden links:* Figure 5.12 (left) shows the results for cases where DCF-fixed is unable to resolve the hidden interference. DCF-flex provides significant throughput gains over DCF-fixed (e.g., 63% at 12 Mbps and 56% at 54 Mbps). However, DCF-flex alone is unable to resolve the conflicts for many other links (e.g., 58% of the links at 54 Mbps). CENTAUR is able to resolve all conflicts by virtue of scheduling. Interestingly, DCF-flex performs better than CENTAUR for a certain fraction of links (e.g., 21% of the links at 54 Mbps with median gain of 33%) — variable channel widths help resolve the conflict, allowing the links to transmit simultaneously. FLUID performs the best (median gain of 74% over DCF-fixed) by using scheduling at 40 MHz when it is not possible to resolve conflicts, and using variable channel widths otherwise, to achieve the maximum throughput.

— *Non-conflicting and exposed links:* Here, CENTAUR and FLUID, both exploit the exposed terminals available at 40 MHz in our topologies e.g., at 12 Mbps and 54 Mbps, throughput for 44% and 21% of the links is improved by up to 2× (Figure 5.12 (right)). The median gain for these exposed terminals was 47%. FLUID performs better than CENTAUR, as it exploits the *additional* exposed terminals that arise when using a combination of different channel widths. At 12 and 54 Mbps, the median gain over CENTAUR for these links was 34% and 42%. Figure 5.12 (right) also shows that FLUID is particularly useful at higher rates, as the number of exposed links available when using only 40 MHz are reduced.

— *Rate adaptation:* With rate adaptation, the median throughput gain of DCF-flex over DCF-fixed was 34% across different conflicting link scenarios. FLUID was able to resolve all conflicts, and for exposed links, we observed gains of up to 1.97×, with median gains of 51% over DCF-fixed and 31% over CENTAUR. We note that the gains were much higher in the presence of hidden links when using SampleRate because the links fall back to a lower rate in case of DCF-fixed, while FLUID can continue to operate at a higher rate.

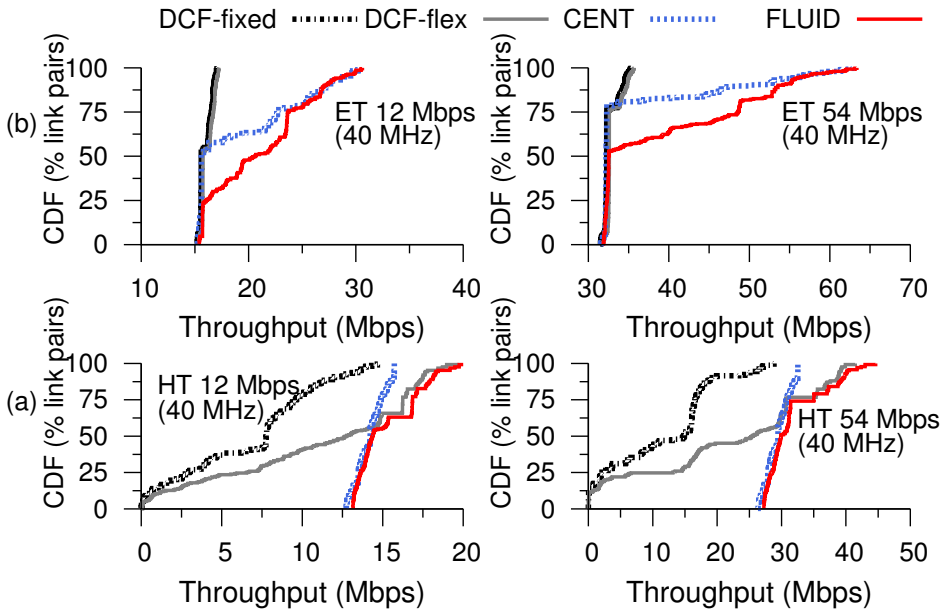

Figure 5.12: The plot shows the CDF of throughputs at 12 and 54 Mbps (40 MHz bandwidth) for two categories: (left) conflicting links or hidden terminals (right) non-conflicting links and exposed terminals. We experimented with 346 two-link topologies for different configurations: DCF-fixed (ii) DCF-flex (iii) CENTAUR and (iv) FLUID.

Performance on a representative topology

We evaluate FLUID on a representative topology by emulating the structure of in-building WLANs. We place our testbed APs near the production APs and clients are randomly distributed into offices without any bias. Our topology consists of 8 APs and 15 clients. For FLUID, we use the modeled conflict graph (§5.4). We also compute the actual conflict graph using bandwidth tests [71] at all possible frequencies and widths. We assume that a total of 40 MHz is available and compare FLUID with DCF-2x20, DCF-40, and CENTAUR. The uplink traffic load is 20% of the downlink. Throughput numbers are averaged over 15 runs. Unless otherwise stated, experiments are run using rate adaptation.

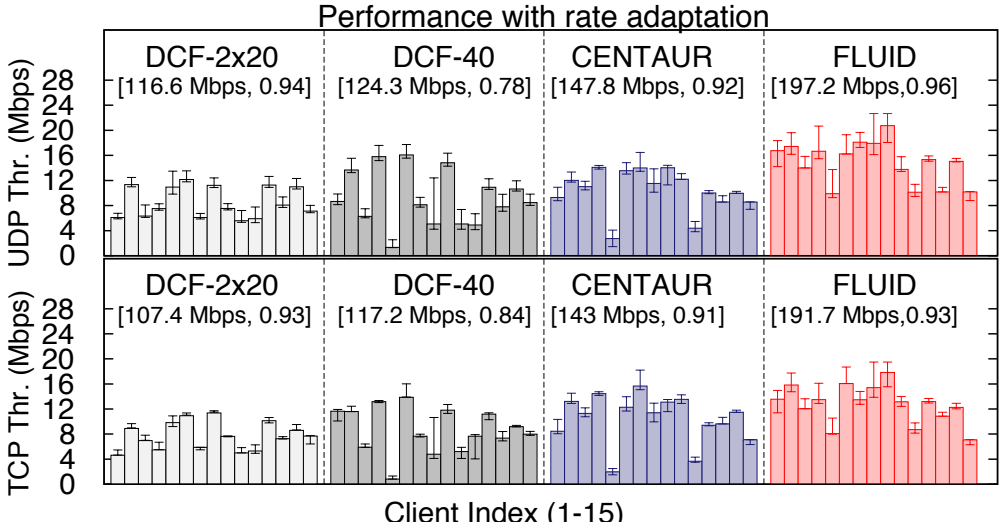

Figure 5.13: Throughput achieved with rate adaptation for a 23 node (8 AP,15 Client) topology. Plot shows the UDP throughput (top) and the TCP throughput (bottom). 10th and 90th percentile values shown by error bars. Sum of values, Jain's Fairness are shown in parenthesis.

— *UDP throughput:* Figure 5.13 (top) shows the UDP throughput for different schemes with rate adaptation. The overall gain was 59% over the best DCF configuration (DCF-40) and 34% over CENTAUR. FLUID significantly improves throughputs of clients which have a lower SNR at 40 MHz (clients 5 and 11), avoiding harmful packing (clients 8 and 9), and increasing transmission concurrency by exploiting partial overlaps and using variable widths whenever possible (e.g., clients 2 and 6). The 10th, 50th, and 90th percentile gains over DCF in this case were 2.66×, 1.54× and 1.21×.

— *TCP throughput:* In this experiment, we run bi-directional TCP traffic with 80:20 downlink/uplink split. Figure 5.13 (bottom) shows the TCP throughputs for each scheme with rate adaptation. We observe that the average gains over DCF-40 and CENTAUR were 63.5% and 34%.

— *Performance of RaC-Pack algorithm:* We evaluated the performance of RaC-Pack algorithm used in FLUID for this topology by comparing it with the brute force approach which picks the 'best' set of schedules, after evaluating

PHY Rate	Gains over best DCF config.			Gains over CENTAUR		
	10th pc	50th pc	90th pc	10th pc	50th pc	90th pc
Fixed 6 Mbps	2.41×	1.62×	1.21×	2.17×	1.31×	1.09×
Fixed 12 Mbps	2.23×	1.63×	1.28×	2.04×	1.34×	1.05×
Fixed 36 Mbps	2.37×	1.57×	1.11×	2.73×	1.46×	1.12×
Fixed 54 Mbps	2.94×	1.71×	1.12×	2.87×	1.58×	1.07×

Table 5.7: Normalized throughput gains of FLUID over the best DCF and CENTAUR across different PHY rates.

all possible schedules offline. We also evaluate the performance of RaC-Pack with the actual conflict graph as input. The aggregate UDP throughputs for these approaches were 206.2 Mbps and 201.4 Mbps respectively, confirming the accuracy of conflict graph and efficiency of the packing approach.

— *Different PHY rates:* Table 5.7 shows throughput gains of FLUID over the best DCF configuration and CENTAUR for different PHY rates. We observe consistent gains across PHY rates (57% - 71% over best DCF, and 31% - 58% over CENTAUR) on this topology as FLUID was able to successfully resolve the conflicts, and exploit the exposed links available with variable channel widths and partial spectral overlaps.

5.8 SUMMARY OF FLUID

In this chapter, we explored the opportunities and challenges in designing 802.11 based wireless LANs employing flexible channelization. We demonstrated that while flexible channelization can improve system throughput, careful construction of flexible channels requires taking into account the interference parameters of the network that depend on the combination of frequencies and channel widths, topology and traffic demand. To this end, we designed and implemented FLUID, a system which improves the throughput of enterprise WLANs by employing joint flexible channelization and data scheduling. Testbed results demonstrate the feasibility of our approach and throughput improvements show that flexible channelization can be a useful parameter in WLAN design.

6 RELATED WORK

In this chapter various ongoing and prior research efforts that are related to this thesis. We first discuss research related to WiFi to WiFi interference detection and quantification. In this portion, we begin by discussing mechanisms that explicitly try to detect, avoid and recover from wireless packet collisions. Next, we discuss proposals that try to model interference between different WiFi links using empirical measurements. Specifically, we focus on mechanisms that try to handle interference in the wireless networks employing flexible channelization. Next, we focus our attention to proposals that try to tackle non-WiFi to WiFi interference. We discuss research prototypes and commercial solutions that detect non-WiFi devices using WiFi hardware or other custom hardware. We then discuss solutions that try to quantify non-WiFi interference and design appropriate mitigation approaches. We close this chapter by discussing efforts in the area of localizing non-WiFi interfering devices.

6.1 HANDLING WIRELESS PACKET COLLISIONS

The problem of handling wireless packet collisions is a fairly difficult one, and there have been a few efforts prior to our COLLIE work that have tried to address this problem. COLLIE was also successful in inspiring a number of subsequent research efforts that have tried to tackle this problem using more sophisticated approaches. We discuss some of these efforts below.

Avoiding collisions. There are have been a number of efforts that focus on *avoiding collisions* in wireless networks. For example, a number of MAC protocols [94] try to avoid collisions by sending control frames or utilizing out-of-band busy tones. One of the popular approaches is to use the RTS/CTS mechanism that avoids some of the collisions. However, it is also well known that such approaches have a high overhead, can result in throughput reduction [21, 22] and hence have been disabled by default in many deployments [31]. Further, compared to COLLIE, these schemes tend to be quite conservative as they try to reserve a large space around communicating nodes [94] leading to

under utilization of the spectrum.

Detecting collisions. Of particular interest is the work that specifically focussed on *detecting collisions* similar to COLLIE. SoftRate [157] utilizes soft physical later hints (SoftPHY hints) to distinguish between losses due to collisions and weak signal for the purposes for rate adaptation. Similarly, AccuRate [144] uses information about constellation dispersions of preamble and postamble to detect packet collisions. In [87], the authors propose a "history" based approach to distinguish between losses due to collisions and weak signal, and adapt rate based only on the errors due to weak signal. Prior to COLLIE, Whitehouse et. al. [160] showed that if two frames arrive at a receiver with certain timing characteristics (the second message arrives after the preamble and start bytes of the first message) and with certain power levels (the second message has significantly higher power level when compared to the first) then it was possible for the receiver to conclude that collision had, indeed, occurred. This mechanism was implemented on the Mica2 sensor mote platform using a 433 MHz Chipcon CC1000 radio transceiver, and required low-level access to timing and signal strength measurements that were available on that platform. In comparison to these works, COLLIE is implemented for off-the-shelf 802.11 wireless transceivers that do not provide such low-level access to communication parameters.

Recovering from collisions. Recently, there has been a growing interest in the wireless networking community to integrate hints from the physical layer, e.g., symbol level information, to solve certain MAC level problems. Some of these efforts have focussed on *recovering from collisions*. Recent examples of works in this category include PPR [74] that proposes a scheme for partial packet recovery using custom error checking, ZipTx [102] and Maranello [61] that make use of known pilot bits to detect errors and recover partial packets. Some of the research endeavors focus on using interference cancellation mechanisms such as SIC [60] and ZigZag decoding [54] that tries to recover packets from repeated collisions. Similarly, ANC [86] is another approach that utilizes knowledge about one of the packets involved in the collision to recover packets. In yet

another work, OFDM symbol dispersions were utilized to present a theoretical framework to distinguish between collisions and weak signal errors in [170]. COLLIE also uses information derived from the physical layer symbols for diagnosing the cause of a packet loss. However, in COLLIE, we work the within the constraints of the commodity WiFi hardware. Specifically, all these approaches require rich information about the physical layer that are not currently exposed by commodity WiFi hardware.

Aborting collisions. There has also been work that focuses on *aborting collisions*. CSMA/CN [143] uses an in-band collision detection mechanisms using self-signal suppression through interference cancellation and antenna orientation. In [119], authors use a very similar concept but propose an out-of-band control channel to transmit pulses to indicate active transmissions and detect potential collisions.

Collision-aware rate adaptation. Some of the rate adaptation mechanisms like RRAA [162] and CARA [21] have, also, tried to address the problem of collision detection in an indirect manner. CARA tries to detect collisions by using the RTS/CTS mechanism, but the proposed mechanism fails in the presence of hidden terminals. CARA also suffers from RTS oscillation [162] which RRAA solves using an adaptive RTS filter. Unlike both RRAA and CARA which try to estimate the collision probabilities by active probing (using an RTS), COLLIE employs a *direct* approach by conducting an empirical post-factum analysis based on receiver feedback. Finally, as mentioned before AccuRate [144] and SoftRate [157] perform explicit collision detection to perform cross layer bit rate adaptation to improve the throughput of wireless links. COLLIE shows similar improvements by enhancing ARF and making it collision-aware. However, it does so by using only information that is exposed by current mainstream wireless cards.

Use of multiple receivers. COLLIE uses *multiple receivers* to improve the accuracy of collision detection. Similar concepts were applied our subsequent work on developing an real time interference estimator, PIE [149] that tries to detect packet collisions across the entire network. Use of multiple receivers

has also been exploited previously in the context of improving throughput in wireless networks, e.g., the Multi-Radio Diversity (MRD) System [109]. More specifically, mechanisms proposed in MRD use multiple receivers to *recover* from bit errors and improve loss resilience, whereas COLLIE uses multiple receivers to determine the *cause of the packet loss* and uses this information for adapting link transmission parameters. Jigsaw [38] also uses information from multiple receivers to provide a global cross-layer viewpoint for enterprise wireless network management.

6.2 MODELING WIFI INTERFERENCE AND FLEXIBLE CHANNELIZATION

We now discuss some of the related work in the area of modeling wireless interference, and focus on some of the work related to flexible channelization. FLUID was the first piece of work that tried to model the interference effects when operating links on flexible channels. Prior to FLUID, most of the work has focussed on modeling WiFi interference on fixed-width networks, and mitigation mechanisms that use flexible channelization for managing wireless interference. We discuss some of these efforts in this section. Since FLUID also employs a scheduling component to enhance the effect of interference mitigation and explore additional transmission opportunities, we also consider some of the scheduling based interference mitigation approaches in this section. Below, we mention these related approaches for tackling WiFi interference and explain how they differ from FLUID.

Modeling WiFi to WiFi interference. There is a large body of work in characterizing wireless interference, especially in the context of modeling the capacity of wireless networks in presence of WiFi interference [52, 53, 57, 58, 85, 93, 95, 99]. Most of these approaches, while useful in providing theoretical insights for the target setting under consideration, make some simplifying assumptions (e.g., assuming a binary interference model) to make the problem more tractable. An alternative body of work [18, 19, 39, 40, 85, 89, 104, 114, 117, 121, 133, 145, 158] has focused on using empirical measurements to provide results that are more helpful in measuring the interference in realistic

settings. Among these works, several systems and models have tried to capture interference between WiFi links using the concept of a "conflict graph". Conflict graph of a wireless network is a graph in which nodes represent the links in a wireless network and edges corresponding to their interference relationships. Research efforts have focussed on deriving the conflict graph of a wireless network either using empirical measurement based approaches, or using modeling procedures, or in some cases a combination of both approaches has been used. We now discuss these efforts below.

Bandwidth test mechanism [117] uses a measurement intensive approach to derive the conflict graph of the network. The basic idea is to let pairs of links simultaneously transmit for a given duration and then measure the amount of throughput reduction observed compared to the throughput observed in the case where links ran in isolation. This work introduced the notion of continuous interference model, wherein the amount of interference (in the range of 0 to 1) depends on the amount of throughput reduction experienced by the links in presence of interference. The work in [114] extends this notion by observing that interference is additive in nature and computes the conflict graph of a network in $O(N^2)$ measurements.

Through some custom firmware modifications, some systems have been able to measure the conflict graph of the network more efficiently. Smarta [19] proposes using micro-experiments, each lasting less than a millisecond, to measure the conflict information of a WLAN. CMAP [158] proposes a mechanism to build the conflict graph of the network on the fly by observing the packet reception probability and disabling carrier sensing opportunistically.

Alternatively, PIE [149] uses a passive mechanism to derive the conflict graph of the network, using concepts similar to the multi-AP collision detection introduced in COLLIE. Several other passive mechanisms [38, 39, 123, 136] have been also been proposed in the literature. These mechanisms use network-wide deployments [38, 39], or state-machine based learning approaches [123, 136] to draw interesting inferences about the network performance and derive the interference relationships between the links in the network.

Finally, several approaches have used explicit modeling mechanisms and analytical techniques to reduce the overhead of conflict graph measurements.

Reis et al. [133] propose a model to capture the conflict graph of the network using pair-wise signal strength models that reduces the overhead to $O(N)$ measurements. The key idea here is to use SINR based models to infer the deferral probability and collision probability of nodes in the network. The work in [84] and [121] extend this model to handle the case of multiple interferers using analytical techniques.

We note that all these methodologies to generate conflict graphs are not suited for use with flexible channelization as they incur significant overhead (Chapter 5). FLUID builds upon some of the the above modeling approaches, and uses signal strength based delivery models to predict interference between links using variable channel widths while allowing arbitrary spectral overlaps.

Flexible channelization. Several channel assignment mechanisms have proposed for both centralized [107] and distributed settings [23, 108]. These mechanisms have been used to mitigate RF interference, and improve fairness along with network throughputs. Existing work on efficient channelization mostly considers 'fixed' width channels [23, 106, 107, 108, 138], and has shown usefulness of client feedback [107], traffic-awareness [138], hopping mechanisms [23, 108, 135] and partial overlaps [106] in this context. FLUID builds upon a number of these ideas, and applies them to design a WLAN employing flexible channelization. In particular, we extend the model presented in [106] (evaluated mostly using simulations) to accommodate variable channel widths and evaluate its accuracy using large scale measurements on commodity hardware.

Recently researchers have explored mechanisms that assign fine grained spectrum blocks on the basis of traffic demands. However, most of these mechanisms, such as KNOWS [169], DSAP [125], Jello [97], WhiteFi [24] and DIMSUMnet [32] are designed for non-802.11 systems (mainly cognitive radios operating over UHF whitespaces) and focus on deriving elegant algorithms for spectrum allocation that are primarily evaluated using network simulations or small-scale prototype implementation on software defined radios.

WhiteFI [24] proposes mechanisms to detect variable channel width transmissions over UHF white spaces. SWIFT [125] is a distributed wideband

spectrum access system that can operate on large frequency bands, even in presence of narrowband signals. Jello [97] proposes using frequency-agile radios to support low latency media applications. Jello focuses on addressing spectrum sensing and fragmentation issues that arise in a distributed setting. Such fragmentation issues in FLUID are minimal, as the controller constructs the flexible channels centrally, and recalculates this assignment every epoch, based on the traffic demand.

Further, FLUID in contrast to the above systems, is a solution that is stylized to 802.11 standard in which the rules of carrier sensing define interference in a certain way leading to hidden and exposed terminals. Moreover, FLUID is evaluated using large scale experiments on a 50 node in-building wireless testbed equipped with off the shelf 802.11 hardware.

Another mechanism that facilitates fine grained spectrum allocation is Orthogonal Frequency Division Multiple Access(OFDMA) [13]. Under OFDMA, different sub-carriers can be assigned to different transmissions, providing robustness against interference. However, as noted in [36], the gains from OFDMA are complementary to that achieved using flexible widths and hence it could be combined with FLUID to provide further improvements.

In the context of 802.11 systems, recent work [36] has shown how adapting channel widths can be beneficial when considering a single, isolated link. Using analysis and simulations, authors in [110] show how channel widths can be used for load balancing. FLUID builds upon a number of these ideas, and extends them significantly to build a practical system capable of leveraging variable width gains under large scale realistic wireless settings. Another approach to changing channel widths is by adding and removing OFDM sub-carriers, as in S-OFDMA. We note that our techniques are useful in such networks as well. If the wireless cards emit the same amount of energy irrespective of the width, then our models hold as is. If the energy varies with the number of sub-carriers, then FLUID's signal interpolation model can be easily modified to scale the transmitted signal with the channel width. Further, FLUID is complementary to recently proposed fine grained frequency division mechanisms like OFDMA [13] and FARA [124], and can be combined with such mechanisms to provide further increase in throughput gains.

Scheduling in enterprise WLANs. Researchers have thoroughly studied scheduling based channel access in wireless networks, and consequently a large body of work has looked into mechanisms for efficient packet scheduling [17, 59, 83, 103, 105, 113, 118, 127, 128, 148, 151, 155, 165]. Below we elucidate on some of these proposals.

Some of the earliest works in the wireless scheduling focused on providing fair proportion of bandwidth to different flows in the network. A distributed fair scheduling algorithm for WLANs was proposed in [155]. A similar scheduling proposal for multi-hop wireless environments was proposed in [83]. TBR (Time Based Regulation) [151] proposed using scheduling to solve rata anomaly issues in 802.11 networks. Other mechanisms have focussed on using frame size modifications [17, 165] to allow time-based fairness in presence of rate diversity. More recently, scheduling has also been proposed in the context of vehicular networking environments [59] to improve client performance. Scheduling has also been proposed in wireless mesh networking scenarios [51, 79] and long distance wireless links [113, 118, 127, 128].

We now discuss some of the more relevant scheduling mechanisms proposed in the context of enterprise WLANs to improve overall network throughput. CENTAUR [148] proposes using epoch-based scheduling mechanism for enterprise WLANs. Specifically, it exploits such scheduling mechanisms to resolve the exposed terminal conflict. FLUID also uses similar epoch-based scheduling mechanisms to allocate flexible channels on a per-epoch basis. More recently, centralized scheduling has also been used in the context of directional antennas in DIRC [103] and MIM-aware transmission re-ordering in Shuffle [105]. These proposals also use a two-phase TDMA approach similar to FLUID to accommodate uplink traffic.

6.3 NON-WIFI DEVICE DETECTION

We now discuss some of the related work in the area of non-WiFi device detection.

Signal classification mechanisms. There is a large body of literature on signal classification that includes work on cyclostationary signal analysis [159],

blind signal detection [101, 115], and other spectrum sensing techniques [20]. Recently, some of the proposals [16, 43] have used neural network classifiers with cyclostationary features to detect the type of modulation used in a received signal. Finally, some of the recent work has also implemented cyclostationary techniques on the USRP platform [24, 78, 116] and evaluated its effectiveness for signal detection and rendezvous in cognitive networks. In contrast to such methods, in our work, we only focus on signal detection methods that can be implemented on top of the functionality exposed by commercial WiFi cards.

Commercial solutions. Present day solutions that detect RF devices include entry-level products like AirMedic [2], Wi-Spy [12] that use extra hardware to display spectrum occupancy, but cannot detect RF devices automatically. More expensive solutions like Cisco Spectrum Expert/CleanAir [5], AirMagnet Spectrum XT [2], and Bandspeed AirMaestro [3] use specialized hardware (signal analyzer ICs) to perform high resolution spectral sampling and detect RF devices. Airshark offers a similar performance, is more cost effective as it operates using commodity WiFi cards, and requires only a software upgrade to be readily integrated in existing WLAN deployments.

Recent research prototypes. Recent research work [66, 98] also leverages specialized hardware to detect non-WiFi devices. Hong et. al [66] use a modified channel sounder to sample a wideband (100 MHz), and present novel cyclostationary signal analysis to accurately detect non-WiFi devices. RFDump [98] uses GNURadio and employs timing/phase analysis along with protocol specific demodulators to detect devices. Airshark builds such functionality under the constraints of using commodity WiFi hardware.

Device-specific solutions. Using controlled measurements, prior work [26, 55, 100, 135, 146] has studied the impact of non-WiFi devices on WiFi links. Many of them have focused on targeted interference scenarios, e.g., between Bluetooth-WiFi [55] or ZigBee-WiFi [100, 146], and proposed mechanisms for co-existence [37, 76]. Similar to [26, 135], we consider the general problem of non-WiFi interference, but specifically, we focus on the problem of making existing WiFi links better aware of non-WiFi RF devices, thereby paving the way

for corrective actions that can be implemented in today's networks. Further, our mechanism is complementary to the above solutions, and can be used in conjunction to more effectively tackle non-WiFi interference.

6.4 NON-WIFI INTERFERENCE DETECTION AND DEVICE LOCALIZATION

We now present the related work in the areas of non-WiFi device interference estimation and localization.

Non-WiFi interference estimation. As mentioned before, commercial solutions such as Wispy [12], Cisco Spectrum Expert [5] and Bandspeed AirMaestro [3] use custom hardware (signal analyzer ICs) to detect RF devices operating in the medium. However, these solutions do not provide the capability to estimate the interference caused by the non-WiFi devices to the the WiFi links. Recent research work such as DOF [66], RFDump [98], TIMO [139] can also detect the presence of non-WiFi device activity using specialized hardware such as channel sounders and software-defined radios. Such platforms enable TIMO and DOF to go beyond detection and employ signal processing techniques to mitigate interference and develop mechanisms to co-exist with non-WiFi devices. WiFiNet takes a step towards empowering APs and clients with such functionality, by providing non-WiFi interference estimation capability under the constraints of commodity WiFi hardware. In [77], the authors use a single WiFi card to infer interference from Bluetooth and microwave ovens by analyzing the timing of WiFi packet errors. However, their technique does not generalize to detect inteference other non-WiFi devices that don't exhibit timing properties (e.g., ZigBee) and cannot distinguish between devices of same type. In comparison, WiFiNet can also estimate the interference from multiple, simultaneously operating devices and pin-point their location in the physical space.

Device localization. There has been limited prior work on designing a generic system to localize the various non-WiFi devices on the top of commodity WiFi hardware. Existing literature has looked at localizing specific device types (e.g., Bluetooth [140], Zigbee [69]) by using sensors of the same type.

Amongst commercial solutions, Wi-Spy device finder [12] uses a directional antenna and requires a user to walk and manually search for the location of the transmitter. Cisco CleanAir [5] finds the location of RF transmitter sources by using specialized hardware in the access points. WiFiNet uses only commodity WiFi cards to not only detect the location of non-WiFi devices, but also estimate their interference impact.

7 CONCLUSIONS AND FUTURE WORK

In this thesis, we have shown how to develop systems and models that help improve our understanding of interference in indoor wireless environments. We have shown how a WiFi AP can differentiate between packet losses due to weak signal and losses due to interference. We have developed systems that further determine whether the losses due to wireless interference are stemming from a non-WiFi interferer or a WiFi interferer. These systems not only detect the wireless interferer, but also quantify the interference impact of the interferer, and physically pin point the location of the interferer. We also explored an alternative approach based on signal strength models that can predict the wireless losses that can happen due to interference between links using flexible channels. Further, all the solutions developed in the thesis have been built on top of commodity WiFi hardware and so they can be natively incorporated into today's WiFi APs and clients to perform wireless loss diagnosis.

We now highlight the main contributions of this thesis in the next section, and then present problems for future research endeavors.

7.1 CONTRIBUTIONS

Below, we list the main contributions of this thesis:

Distinguishing between weak signal and collisions: Our work demonstrated that the inability to distinguish between losses due to collisions and weak signal in current 802.11 systems has led to conservative design of assuming collision as the default cause for packet loss and that it translates to wasted bandwidth, energy and significant performance degradation. To address these issues, we developed the first collision inferencing engine, COLLIE, that works on top of mainstream wireless cards. Specifically, we developed (i) algorithms that expose statistical differences between collision and signal degradation based losses through empirical analysis; (ii) a protocol that capitalizes on the judgment from the algorithms by aptly adjusting the correct link-level parameters for 802.11.

Our evaluation results demonstrated that COLLIE can provide up-to 95% accuracy in detecting collisions while allowing a configurable false positive rate of 2%. We showed that rate adaptation mechanisms such as ARF (auto-rate fallback) when made collision-aware can lead to throughput improvements between 20- 60%. Through an emulation of voice call (made using the Netgear SPH101 Voice-over-WiFi phone), we also showed that our collision inferencing mechanisms help reduce retransmission related costs by 40% for different mobility scenarios. This work was successful in highlighting a fundamental drawback in 802.11's design and has inspired significant follow-on work [42, 88, 90, 91, 96, 142, 143, 144, 157] in the wireless networking community.

In summary, this was the first piece of work that has shown how to design collision inferencing mechanisms purely on top of commodity WiFi cards, and how such inferences can feed into existing mechanisms such as rate adaptation to provide significant throughput gains in wireless networks.

Detecting non-WiFi interferers using WiFi hardware: We carried out the first measurement study to characterize the prevalence of non-WiFi RF devices in typical environments such as homes, offices, and various public spaces. Using data corresponding to more than 600 hours (spanning around 6 weeks and across 21 locations) collected using signal analyzers we established the prevalence of non-WiFi devices — these devices are popular across many locations, and often appear with high signal strength. We then designed Airshark, a software system that runs on top of commodity WiFi hardware and can detect the presence of various non-WiFi devices. Airshark makes use of the fine-grained spectrum information provided by emerging WiFi cards, and utilizes machine learning algorithms to detect a device's presence.

Our evaluation showed that Airshark can detect multiple non-WiFi devices including fixed frequency devices (e.g., ZigBee, analog cordless phone), frequency hoppers (e.g., Bluetooth, game controllers like Xbox), and broadband interferers (e.g., microwave ovens). Airshark has an average detection accuracy of 91−96%, over a wide range of signal strengths (−80 to −30 dBm) and has a low false positive rate (0.39% on an average). Further, Airshark's performance was comparable to commercial signal analyzers that employ custom hardware.

In summary, this work has shown how to design a generic system to detect non-WiFi RF devices using only commodity WiFi cards so that WiFi APs and clients to natively perform such detection in today's WLANs without any additional hardware.

Quantifying and localizing non-WiFi interference: We designed WiFiNet, the first software system that can quantify and localize interference experienced by WiFi links in presence of non-WiFi devices using only commodity WiFi hardware. WiFiNet utilizes tight clock synchronization among WiFi APs and employs signal clustering techniques operating on some non-WiFi device specific attributes (when available) and signal strength observations gathered by multiple WiFi APs to identify the unique transmission contributions from different, potentially identical, non-WiFi devices. By doing so, WiFiNet correctly estimates the impact of each non-WiFi device, in presence of multiple other interferers, even if they are of the same type. Our evaluation showed that WiFiNet's interference estimates are within $\pm 10\%$ of the ground truth. WiFiNet also correctly tracks changes due to client mobility, dynamic traffic loads, and varying channel conditions. We also showed that WiFiNet can identify the physical locations of non-WiFi devices accurately. Our evaluation experiments showed that median localization error was $\leqslant 4$ meters. Further, WiFiNet can also be extended to quantify interference from WiFi devices.

In summary, WiFiNet is the first system that can detect, quantify and localize the interference from non-WiFi interfering devices (as well as WiFi interferers) in real-time and using commodity WiFi hardware alone. Such a system enables WLAN administrators to use commodity WiFi APs to better understand and manage WiFi and non-WiFi interference, especially in enterprise WLANs.

Modeling wireless interference in the context of flexible channelization: We demonstrated that it is possible to model the wireless interference (WiFi to WiFi interference) in presence of multiple WiFi links employing flexible channelization — the choice of an appropriate channel width and center frequency for each transmission. Our wireless interference models for flexible channelization are based on empirical signal strength measurements and they take into account that the interference properties of the network that depend

on the combination of frequencies and channel widths, topology and traffic demand. We designed and implemented FLUID, a system which improves the throughput of enterprise WLANs by employing joint flexible channelization based on our interference models. Through FLUID, we showed how enhancing flexible channelization with data scheduling can maximize the number of simultaneous transmissions and hence improve throughput.

We implemented FLUID in an enterprise-like setup using a 50 node testbed with off-the shelf wireless cards. Our evaluation of FLUID's interference models demonstrated a prediction accuracy of 87%. We showed that FLUID improves the average throughput by 59% across all PHY rates, compared to existing fixed-width approaches.

In summary, we showed how to model WiFi to WiFi interference in systems employing flexible channelization using empirical signal strength based models. Our results demonstrated the feasibility of our modeling approach and throughput improvements showed that flexible channelization interference models can be used as a crucial input in WLAN design.

7.2 KEY TAKEAWAYS AND LESSONS LEARNT

We now present some of the key takeaways from this dissertation work and discuss the lessons learnt along the way. We then describe the features lacking in current WiFi cards, and as a feedback to WiFi chipset designers, we describe some of the desirable features in future wireless cards that would help us build systems that can better estimate interference in indoor wireless environments.

Key takeaways: We first discuss the key lessons learnt from our work on WiFi to WiFi interference estimation (COLLIE and FLUID).

- Distinguishing between errors that happen due to collision and weak signal is important as being agnostic to the type of errors leads to wasted bandwidth, energy and significant performance degradation.
- Using limited information provided by today's WiFi cards (e.g., RSSI and failure bit error patterns), it is possible to design algorithms that can

achieve a collision detection accuracy of up to 95%. Further, making link adaptation algorithms collision-aware improves throughput up to 30%.

- Changing channel widths on links using commodity WiFi hardware results in changing the interference relationship between these links. Further, it is possible to model such interference effects using RSSI information provided by commodity WiFi cards. The median accuracy of our models was 88%.

We now discuss the key lessons learnt from our work on non-WiFi to WiFi interference estimation (Airshark and WiFiNet).

- Non-WiFi interference is prevalent in today's indoor wireless environment, and non-WiFi devices lead to significant throughput degradation (more than 50% in many cases) of good quality WiFi links.

- While current WiFi cards only provide limited signal information (e.g., RSSI per sub-carrier), it is possible to detect a limited number (up to 6 in our experiments) of simultaneously operating non-WiFi devices using such information. We note that this information about the received power is common to any wireless technology — *fundamentally, every wireless technology employs radio emissions that inherently consist of electro-magnetic energy (or power). Therefore, a solution that is developed solely on top of such received power is generic, and is potentially applicable to all wireless technologies*. To our advantage, the very fact that different radio technologies employ diverse physical layer and MAC layer mechanisms (e.g., protocols, channel access methods, modulation mechanisms) can be used to detect such devices — *different radio technologies exhibit different transmission power patterns in the spectrum that can be used to uniquely identify them*. The average accuracy of our detection algorithms based on such power patterns was 91-96% even in the presence of multiple simultaneously active RF devices operating at a wide range of signal strengths (-80 to -30 dBm).

- Combining such received power patterns from multiple WiFi cards, it is possible to design systems that can also distinguish between multiple

devices of the same type (e.g., two cordless phones of the same model), estimate the individual interference impact of each such non-WiFi device and even physically pin point the location of each device. Experimental results show that interference estimates were within $\pm 10\%$ of the ground truth and the median localization error was $\leqslant 4$ meters.

Feedback to wireless chipset designers: Based on our experience in building interference estimation systems that make use of today's WLAN hardware, we now discuss some of the desirable features and capabilities in future wireless cards. We expect these new features and capabilities to be useful in designing software systems that run on top of WiFi cards and can further improve the accuracy of interference estimation in indoor WLANs. Below, we first present features that may be useful in improving the accuracy of WiFi to WiFi interference estimation and argue for their utility:

- While information about RSSI per packet was helpful in identifying whether a packet was received in error due to collision or weak signal, the accuracy of our collision detection algorithms suffered in some cases (e.g., due to capture effect). We believe this is partly because a metric like RSSI, while useful, is not precise for the purposes of collision detection — RSSI is calculated during the preamble stage of receiving an 802.11 frame and thus cannot capture the spike in the received power due to interference. Newer metrics such as Received Channel Power Indicator (RCPI) that measure received RF power in a selected channel over the preamble and the entire received frame would be more useful in this context. However, RCPI is not exposed by current WiFi hardware. Of particular interest is provision for fine-grained information about RSSI (averaged over a few OFDM symbols) that can enable better detection of collisions — a sudden spike in the instantaneous RSSI would then enable easier collision detection using commodity WiFi cards [156].

- Metrics such as symbol error rate were also useful in our collision detection algorithms. However, we could not obtain precise information about the actual physical layer bits that were received in error. This

is mainly because of procedures such as scrambling, interleaving and convolution coding are applied to the data bits to output the physical layer bits. Providing a limited amount of additional information (e.g., scrambler seed used) can help determine the exact physical layer bits in error as this mapping (scrambling, interleaving and convolution coding) is deterministic and reversible. This would enable us to use novel metrics based on actual physical layer error bits to improve the accuracy of collision detection.

- Alternative to the above, exposing some additional information about the physical layer decoding process in commodity WiFi cards — actual bits received in error, measured constellation dispersions, and log likelihood ratios (LLRs) of the decoders — is shown to be useful in increasing the accuracy of collision detection [143, 144, 156].

Most of the inaccuracies in detecting non-WiFi devices using today's WiFi cards arose from reduced sampling quality attributable to a variety of reasons. We now describe these reasons, and argue for inclusion of additional features and capabilities that may be useful in improving the accuracy of non-WiFi to WiFi interference estimation.

- Information about the received signal can be exposed at various levels of accuracy. For example, software radios are capable of exposing baseband I/Q samples (output by the ADC) that precisely represent the signal information. Wireless cards can also expose such precise information enabling us to develop custom software modules to not just detect a particular non-WiFi device, but be able to decode its transmission. This would not only allow for interference detection and avoidance, but also help pave way for developing cross-technology co-existence mechanisms. Such an approach, however, requires high processing capabilities at the AP or the client using this WiFi card. At the other end of spectrum is exposing only RSSI (averaged across an entire packet), or RSSI per sub-carrier (averaged across a few OFDM symbols). While this information does not allow us to decode the signal in its entirety, it allows us to detect

different devices. On the plus side, this requires minimal processing capabilities on the AP (or client). Further, sampling rate can also be controlled (based on the number of symbols used for averaging) providing us with different amounts of precision. We recommend that wireless cards should allow for a flexible sampling approach, one that enables network administrators or wireless vendors to determine the precise accuracy (raw I/Q samples vs. RSSI per sub-carrier for every OFDM symbol vs. RSSI per sub-carrier averaged for a few symbols) of these measurements based on the wireless environment, device's processing capabilities and energy constraints. Such a flexible approach would allow for opportunistic mechanisms e.g., richer information can be requested for dense wireless environments that host multiple non-WiFi devices, whereas information at a coarser granularities can be used in relatively quiet environments to conserve processing and energy costs.

- Current 802.11 a/b/g/n cards can only sample a limited amount of spectrum bandwidth at any given instant. This leads to missing samples from other parts of the spectrum. For example, today's WiFi cards can only capture samples from a spectrum bandwidth of either 20 MHz or 40 MHz, whereas the spectrum of interest could be much wider, as in the case of 80 MHz wide 2.4 GHz band. We recommend that WiFi cards should potentially be able to sample wider bands of spectrum — non-WiFi interference solutions running on top of WiFi cards that can potentially sample larger widths of spectrum will have improved accuracy and faster convergence times. For this to happen, crystal oscillators in WiFi cards that determine the clock speed[1] should be able to run at a higher speed of 80 MHz, as opposed to today's WiFi cards that support speeds up to 40 MHz. We note that such faster clocks speeds are possible in future wireless cards. As an example, emerging technologies like 802.11ac can have communication channels as wide as 160 MHz, thus requiring support for

[1]This clock is eventually fed into the frequency synthesizers and PLLs that determine the center frequency and bandwidth of the WiFi card respectively. For example, a PLL clock speed of 40 MHz corresponds to a bandwidth of 40 MHz [36].

faster clock speeds. Running our non-WiFi detection algorithms on such future wireless cards supporting 802.11ac should have better performance.

- While increasing the sampling bandwidth of WiFi cards is useful, generally, one can expect that the amount of spectrum that has to be monitored for non-WiFi device activity might be more than the sampling bandwidth. In such cases, non-WiFi detection mechanisms such as Airshark would require the WiFi card to *switch* to different parts of the spectrum and gather samples. We find that channel switching times for current WiFi cards are rather large — in our experiments, the median time to switch the channel was around 19.7 ms. Such high switching times result in missing samples and hence lead to detection inaccuracies. We recommend that the channel switching times in the current WiFi cards be improved. Switching times of WiFi cards are typically determined by the time taken for frequency synthesizers to settle [141]. Improvements in technology leading to faster settling times for the frequency synthesizer would reduce the number of missing samples and potentially improve detection capabilities of our algorithms.

- Currently, in WiFiNet, only information from APs operating on the same channel can be combined to detect non-WiFi devices (operating in that channel) and estimate their interference impact on WiFi links using the same channel. Concepts used in WiFiNet can be extended to a scenario where multi-channel monitoring capability is needed. A limiting factor in our current system is to be able to synchronize APs running on different channels — currently, WiFiNet uses common WiFi packet receptions to determine the clock skews among different APs operating on the same channel, and then runs a time synchronization algorithm. We recommend that chipset manufacturers provide primitives that facilitate synchronization among clocks on WiFi cards that potentially operate on multiple channels. Using today's WiFi cards, multi-channel synchronization can be enabled only by deploying multiple radios on each AP and using a single local clock to timestamp the received packets [38].

- Lastly, it is desirable that a global repository of non-WiFi devices and their transmission characteristics (or signatures as described in Chapter 3) be maintained. Unlicensed spectrum is a host to a plethora of non-WiFi devices that employ a variety of proprietary technologies. Maintaining a single repository that maps the devices (e.g., using their FCC IDs) to their signatures would enable re-use of these signatures and obviate training stage of Airshark. However, we note that such signatures would also have to take the wireless card under consideration — appropriate normalization procedures in combination with WiFi card specific thresholds (See Chapter 3) would have to be employed.

Summary and future outlook. In summary, current WiFi hardware provides limited amount of information about the received signal such as power characteristics. We find that such information is useful in a number of cases including collision detection, non-WiFi device detection and non-WiFi interference estimation. Moreover, these approaches are generic, and we expect them to be applicable to future wireless technologies. This is because, while information provided by current WiFi cards is limited, such information is fundamental to all wireless technologies — every wireless technology, irrespective of the physical and MAC layer mechanisms under consideration, employs radio emissions that inherently consist of electro-magnetic energy (or power). Thus a solution that is developed solely on top of such received power is generic, and is potentially applicable in context of future wireless technologies such as 802.11ac, and enables them to detect a variety of newer wireless devices such as motion detectors, car alarms, lighting and HVAC controls that have started to proliferate in the unlicensed band [139].

However, we note that the utility of this information (*i.e.*, only received power values) is also limited in some cases. For example, while it is possible to detect a small number ($\leqslant 6$) non-WiFi devices that are active simultaneously, detection approaches might not work well when increasing the number of active devices or when detecting devices at low SNR. This is probably not a concern in today's WLANs where a maximum of 3 to 4 non-WiFi devices happen to be active simultaneously (See Chapter 3). In future, however, as the

number of non-WiFi devices proliferate, we expect that some of the additional desirable features mentioned above would be helpful in improving the accuracy of detection mechanisms that work on top of commodity WiFi cards.

7.3 FUTURE WORK

We believe that this dissertation was successful in developing some of the important building blocks to improve our understanding of wireless interference in indoor wireless environments. We now describe some of the potential problems for future research in this domain.

Leveraging client input for improved interference quantification: Interference quantification methods used in WiFiNet only use the input (observations about WiFi and non-WiFi device transmissions) from WiFi APs. Enhancing WiFiNet using the feedback from WiFi clients can improve the system in a number of ways: (a) aggregating WiFi client transmission reports can help WiFiNet also estimate *uplink interference* in the network. Such information would be particularly useful for AP-client links running bi-directional, interactive applications such as video conferencing or VoIP applications, (b) using clients to report other WiFi transmissions can help improve WiFiNet's coverage — WiFi transmissions from other APs and clients not accounted before can help improve overall accuracy for WiFi to WiFi interference quantification, (c) clients with compatible WiFi hardware might also be able to run Airshark, thus enabling them to detect additional non-WiFi devices in the client's vicinity thus improving WiFiNet's coverage, accuracy of non-WiFi to WiFi interference quantification as well as non-WiFi device localization accuracy.

It is important to note, however, that using client feedback also presents us with some new challenges in the design of WiFiNet: (a) client devices (e.g., WiFi enabled laptops, tablets and smartphones) are typically energy constrained whereas running Airshark would require the WiFi card to be continuously on thereby implying that client devices might not be able to use WiFi's power-save mode effectively. (b) Depending on the type of coverage required, the network administrator might want WiFiNet to report interference information about

multiple WiFi channels. WiFiNet leverages an extra radio on the WiFi AP to continuously switch channels and aggregate interference information. Such a functionality, however, might not be desirable on the clients as it might disrupt existing AP-client communication.

WiFiNet's design would have to take these additional constraints into account before scheduling tasks such as non-WiFi device detection, WiFi transmission sniffing etc. on client devices.

Enhancing interference quantification through active measurements: Airshark andWiFiNet are systems that use *passive* observations about device transmissions to detect, quantify and localize wireless interference. An alternative method is to estimate the impact of interference through limited amount of *active* measurements. For example, detecting a non-WiFi device activity, WiFiNet controller can schedule transmission probes for WiFi APs (and clients) in the vicinity of the non-WiFi device to quantify the device's impact — scheduled probes would not only provide faster convergence of interference estimates, but might also improve WiFiNet's accuracy of interference estimation due to reduced "overlapping transmissions" from WiFi devices. Such a design, however, has to take additional parameters such as the overhead of probe traffic into account.

Designing better algorithms for interference estimation and localization: While the accuracy of device detection, interference quantification and device localization are quite reasonable, they can be further improved in a number of ways. For example, Airshark's device detection accuracy can be improved by exploring additional features and more complex machine learning algorithms that can better handle the cases where the transmissions of non-WiFi devices always overlap (Chapter 3). There is also a scope for designing better metrics using concepts from information theory such as J-score [82] that also take the frequency of an interference event into account. Device localization methods used in WiFiNet can also be improved by utilizing more complex propagation models [65, 75, 120], richer information about the deployment [75] and alternative algorithms [41] into account.

Extending interference quantification for other wireless environments: Designing interference quantification mechanisms in the context of home WLANs is an important research problem as it is well known that home WLANs are typically prone to interference from a variety of WiFi and non-WiFi devices [131]. While the systems developed in this thesis were designed for, and evaluated on enterprise WLAN settings, some of the concepts used in this systems are also applicable to home wireless LANs. For example, COLLIE's collision detection mechanisms that are based on analysis of bit-error patterns and Airshark's device detection capabilities are applicable in the context of a single WiFi node (or a single WiFi link) systems typical of home WLANs. Availability of multiple WiFi APs (and clients) can be used to further improve the accuracy of these mechanisms. However, systems such as WiFiNet, and FLUID, are based on the assumption that information is aggregated from multiple vantage points (APs and clients). Such an assumption might not be suitable for home WLANs. Further, the lack of a central control element (e.g., a WLAN controller) makes the problem a lot more challenging. It remains to be seen whether mechanisms similar to WiFiNet can be designed with the help of a single WiFi AP that uses non-WiFi device transmission reports from Airshark and still be able to infer lost transmissions [123] to perform transmission overlap analysis [132] and quantify the interference impact in home WLAN environments.

Extending interference models to account for non-WiFi interference: FLUID's interference models only take WiFi to WiFi interference into account. Similar interference models can be developed to take non-WiFi device interference into account. For example, information from Airshark can be used to capture non-WiFi device's signal strength information, and using information about device's power spectral density, models similar to those used in FLUID that take the spectral overlap [106] into account can be developed. Such models can provide information about non-WiFi device's interference impact before the event of a packet loss and help APs and clients take proactive approaches to avoid interference.

A IMPACT OF THIS DISSERTATION

We briefly summarize the publications and the broader impact of this thesis below.

- **COLLIE**: COLLIE was the first prototype running on top of mainstream WiFi cards that can discern whether a packet loss was due to a collision or due to weak signal. COLLIE was published in Infocom 2008 [130], and it also won the ACM student research competition at MobiCom 2007. COLLIE was successful in highlighting a fundamental drawback in 802.11's design and has inspired significant follow-on work in the wireless networking community [42, 88, 90, 91, 96, 142, 143, 144, 157][1]. Our own enhancements to the multi-AP assisted collision inferencing mechanisms introduced in COLLIE resulted in a system called PIE [149]. PIE can quantify both transmitter-side (carrier sensing) and receiver-side (e.g., collision) interference for links across an entire WLAN in real-time, and with no wireless measurement overhead. PIE was published in NSDI 2011.

- **Airshark**: Airshark was the first research prototype that uses only off-the-shelf WiFi cards to detect the presence of non-WiFi wireless devices in real-time. Airshark was published in IMC 2011 [131] and has been featured in many news outlets (e.g., *Slashdot, NetworkWorld, CRA research highlight*). Recently, several wireless vendors such as Aruba [10] have also started working on similar solutions that detect non-WiFi devices using WiFi cards.

- **WiFiNet**: WiFiNet was the first system that detects, quantifies and localizes interference impact of various non-WiFi sources using commodity WiFi hardware alone. WiFiNet was published in NSDI 2012.

- **FLUID**: FLUID showed that the interference relationships between WiFi links depend on the channel width of operation. Combined with data

[1] As of May 2012, a total of 71 research papers have cited COLLIE according to Google scholar

scheduling, FLUID showed that significant throughput gains can be achieved in a enterprise WLAN setting. FLUID was nominated for the best paper award at MobiCom 2011 and was one of the three papers fast-tracked to IEEE Transactions of Mobile Computing.

[1] Agilent spectrum analyzers (signal analyzers). http://www.agilent.com/.

[2] AirMagnet AirMedic and Spectrum XT. www.airmagnet.net/products.

[3] Bandspeed AirMaestro spectrum analysis solution. http://www.bandspeed.com/.

[4] Cisco centralized wlan solution: Overview. `http://www.cisco.at/reseller/2005_08/Aironet_Controller.pdf`.

[5] Cisco Spectrum Expert. http://www.cisco.com/en/US/products/ps9393/index.html.

[6] Intel pro/wireless network connection for mobile. http://www.intel.com/network/connectivity/products.

[7] LIBSVM: A Library for Support Vector Machines. http://www.csie.ntu.edu.tw/ cjlin/libsvm/.

[8] Madwifi wireless driver. http://madwifi-project.org.

[9] Meru controllers: Five times the capacity of other controllers. Http://www.merunetworks.com/products/hardware/controllers/index.html.

[10] Mobile Edge Architecture of Aruba Wireless Networks. Http://www.arubanetworks.com/technology/mobile-edge/architecture.

[11] The WEKA data mining software: an update. ACM SIGKDD Explorations Newsletter.

[12] Wi-Spy spectrum analyzer. www.metageek.net.

[13] Wimax forum whitepapers. Available at: http://www.wimaxforum.org.

[14] 2007. The Android Operating System. `http://www.android.com/`.

[15] 2007. The Apple iOS Operating System. `http://www.apple.com/iphone/ios/`.

[16] A. Fehske, J. Gaeddert, and J. Reed. 2005. A new approach to signal classification using spectral correlation and neural networks.

[17] A. Munaretto, M. Fonseca, K.A. Agha, and G. Pujolle. 2004. Fair time sharing protocol: A solution for IEEE 802.11b hot spots. In *Lecture notes in computer science*.

[18] Aguayo, Daniel, John Bicket, Sanjit Biswas, Glenn Judd, and Robert Morris. 2004. Link-level measurements from an 802.11b mesh network. In *Sigcomm*.

[19] Ahmed, N., and S. Keshav. 2006. Smarta: A self-managing architecture for thin access points. In *Conext*.

[20] Akyildiz, Ian F., Won-Yeol Lee, Mehmet C. Vuran, and Shantidev Mohanty. 2006. Next generation/dynamic spectrum access/cognitive radio wireless networks: a survey. *Comput. Netw.*

[21] et al., J. Kim. 2006. Cara: Collision-aware rate adaptation for ieee 802.11 wlans. In *Infocom*, 139–150.

[22] Atheros. 2006. âŁœ802.11 wlan peformanceâŁž.

[23] Bahl, Paramvir, Ranveer Chandra, and John Dunagan. . Ssch: slotted seeded channel hopping for capacity improvement in ieee 802.11 ad-hoc wireless networks. In *Mobicom '04*.

[24] Bahl, Paramvir, Ranveer Chandra, Thomas Moscibroda, Rohan Murty, and Matt Welsh. . White space networking with wi-fi like connectivity. In *Sigcomm '09*.

[25] Bahl, Paramvir, and Venkata N. Padmanabhan. Radar: an in-building rf-based user location and tracking system. In *Infocom'00*.

[26] Baid, Akash, Suhas Mathur, Ivan Seskar, Tripti Singh, Shweta Jain, Dipankar Raychaudhuri, Sanjoy Paul, and Amitabha Das. Spectrum MRI: Towards diagnosis of multi-radio interference in the unlicensed band. In *Ieee wcnc 2011*.

[27] Bhargavan, V., A. Demers, S. Shenker, and L. Zhang. 1994. Macaw: A media access protocol for wirelesss lans. In *Proceedings of acm sigcomm*.

[28] Bicket, J. 2005. Bit-rate selection in wireless networks. MIT Master's Thesis.

[29] B.Lin and J. Wu. 2003. Analysis of hyperbolic and circular positioning algorithms using stationary signal-strength-difference measurements in wireless communications. In *Vtc*.

[30] Brik, Vladimir, Eric Rozner, and Suman Banerjee. 2005. Dsap: a protocol for coordinated spectrum access. In *In ieee dyspan*.

[31] Broadcom. 2005. Wireless lan adpater user guide.

[32] Buddhikot, Milind M., Paul Kolodzy, Scott Miller, Kevin Ryan, and Jason Evans. 2005. Dimsumnet: New directions in wireless networking using coordinated dynamic spectrum access. In *Wowmom05 '05*, vol. 1.

[33] Bychkovsky, V., B. Hull, A. Miu, H. Balakrishnan, and S. Madden. 2006. A measurement study of vehicular internet access using in situ wi-fi networks. In *Mobicom*.

[34] Canalys. 2011. Smart phones and pads fuel wireless LAN growth. `http://www.canalys.com/newsroom/smart-phones-and-pads-fuel-wireless-lan-growth`.

[35] Castro, Rui, and Robert Nowak. 2009. Active sensing and learning.

[36] Chandra, Ranveer, Ratul Mahajan, Thomas Moscibroda, Ramya Raghavendra, and Paramvir Bahl. A case for adapting channel width in wireless networks. In *Sigcomm '08*.

[37] Chek, Michael Cho-Hoi, and Yu-Kwong Kwok. Design and evaluation of practical coexistence management schemes for Bluetooth and IEEE 802.11b systems. In *Computer networks,'07*.

[38] Cheng, Y., J. Bellardo, Péter Benkö, A. Snoeren, G. Voelker, and S. Savage. 2006. Jigsaw: solving the puzzle of enterprise 802.11 analysis. *SIGCOMM*.

[39] Cheng, Yu-Chung, Mikhail Afanasyev, Patrick Verkaik, Péter Benkö, Jennifer Chiang, Alex C. Snoeren, Stefan Savage, and Geoffrey M. Voelker. 2007. Automating cross-layer diagnosis of enterprise wireless networks. *SIGCOMM Comput. Commun. Rev.*

[40] Cheng, Yu-Chung, John Bellardo, Péter Benkö, Alex C. Snoeren, Geoffrey M. Voelker, and Stefan Savage. 2006. Jigsaw: solving the puzzle of enterprise 802.11 analysis. In *Sigcomm*.

[41] Chintalapudi, Krishna, Anand Padmanabha Iyer, and Venkata N. Padmanabhan. 2010. Indoor localization without the pain. In *Proceedings of the sixteenth annual international conference on mobile computing and networking*, 173–184. MobiCom '10, New York, NY, USA: ACM.

[42] Cho, Mingu, Hakyung Jung, Shinhaeng Oh, Ted "Taekyoung" Kwon, and Yanghee Choi. 2011. Distinguishing collisions from low signal strength in static 802.11n wireless lans. In *Proceedings of the acm conext student workshop*, 15:1–15:2. CoNEXT '11 Student, New York, NY, USA: ACM.

[43] Davy, Manuel, Arthur Gretton, Arnaud Doucet, Peter J. W. Rayner, and Associate Member. 2002. Optimized support vector machines for nonstationary signal classification. In *Ieee signal processing letters*, 9–12.

[44] Du, Pan, Warren A. Kibbe, and Simon M. Lin. Improved peak detection in mass spectrum by incorporating continuous wavelet transform-based pattern matching. In *Bioinformatics'06*.

[45] Eriksson, Jakob, Sharad Agarwal, Paramvir Bahl, and Jitendra Padhye. 2006. Feasibility study of mesh networks for all-wireless offices. In *Mobisys*.

[46] F. Herzel, G. Fischer, and H. Gustat. An integrated cmos rf synthesizer for 802.11a wireless lan. In *Ieee jrnl. of solid-state circuits'03*.

[47] Fernando, Xavier N., and Balakanthan Balendran. 2005. Adaptive denoising and equalization of infrared wireless cdma system. *EURASIP J. Wirel. Commun. Netw.* 2005(1):20–29.

[48] Future, Mobile. 2011. The 2011 Mobile Year in Review. `http://mobilefuture.org/page/-/images/2011-MYIR.pdf`.

[49] G. Palshikar. Simple algorithms for peak detection in time-series, trddc technical report'09. In *Technical report, trddc*.

[50] G. Stuber. Principles of Mobile Communication.2000.

[51] Gambiroza, Violeta, Bahareh Sadeghi, and Edward W. Knightly. 2004. End-to-end performance and fairness in multihop wireless backhaul networks. In *Acm mobicom*.

[52] Garetto, Michele, Theodoros Salonidis, and Edward W. Knightly. 2006. Modeling per-flow throughput and capturing starvation in csma multi-hop wireless networks. In *Ieee infocom*.

[53] Gastpar, Michael, and Martin Vetterli. 2002. On the capacity of wireless networks: The relay case. In *Proceedings of ieee infocom*.

[54] Gollakota, Shyamnath, and Dina Katabi. 2008. Zigzag decoding: combating hidden terminals in wireless networks. *SIGCOMM Comput. Commun. Rev.* 38(4):159–170.

[55] Golmie, N., N. Chevrollier, and O. Rebala. Bluetooth and WLAN coexistence: Challenges and solutions. *IEEE Wireless Communications Magazine'03*.

[56] Gonzalez, Rafael C., and Richard E. Woods. Digital image processing, 3rd ed. 2006.

[57] Grossglauser, Matthias, and David Tse. 2001. Mobility increases the capacity of ad-hoc wireless networks. In *Proceedings of ieee infocom*.

[58] Gupta, Piyush, and P. R. Kumar. 2000. The capacity of wireless networks. *IEEE Transactions on Information Theory*.

[59] Hadaller, David, Srinivasan Keshav, Tim Brecht, and Shubham Agarwal. 2007. Vehicular opportunistic communication under the microscope. In *Acm mobisys*.

[60] Halperin, Daniel, Thomas Anderson, and David Wetherall. 2008. Taking the sting out of carrier sense: interference cancellation for wireless lans. In *Mobicom*.

[61] Han, Bo, Aaron Schulman, Francesco Gringoli, Neil Spring, Bobby Bhattacharjee, Lorenzo Nava, Lusheng Ji, Seungjoon Lee, and Robert R. Miller. 2010. Maranello: Practical partial packet recovery for 802.11. In *Nsdi*, 205–218. USENIX Association.

[62] Harmer, K., G. Howells, W. Sheng, M. Fairhurst, and F. Deravi. A peak-trough detection algorithm based on momentum. In *Cisp'08*.

[63] Haupt, Jarvis, Rui Castro, and Robert Nowak. 2010. Distilled sensing: Selective sampling for sparse signal recovery.

[64] Henderson, T., D. Kotz, and I. Abyzov. 2004. The changing usage of a mature campus-wide wireless network. In *Acm mobicom*.

[65] Hills, Alex, Jon Schlegel, and Ben Jenkins. 2004. Estimating signal strengths in the design of an indoor wireless network. In *Ieee transactions on wireless communications*.

[66] Hong, Steven, and Sachin Katti. DOF: A local wireless information plane. In *Acm sigcomm 2011*.

[67] IEEE. Wireless lan medium access control (MAC) and physical layer (PHY) spec, IEEE 802.11 standard. *IEEE Standard 802.11*.

[68] Inc., Qualcomm Atheros. 2010. The Atheros AR92xx series datasheet.

[69] J. Blumenthal et. al. Weighted centroid localization in zigbee-based sensor networks. *IEEE ISISP'07*.

[70] J. Han and M. Kamber. Data Mining: Concepts and Techniques.

[71] J. Padhye et al. Estimation of link interference in static multi-hop wireless networks. In *Imc'05*.

[72] J. Sander et. al. Density-based clustering in spatial databases. *Data Mining Knowledge Discovery'98*.

[73] Jain R. et. al. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. In *Tech. report'84*.

[74] Jamieson, K., and H. Balakrishnan. 2007. Ppr: Partial packet recovery for wireless networks. In *Acm sigcomm*.

[75] Ji, Yiming, Saâd Biaz, Santosh Pandey, and Prathima Agrawal. 2006. Ariadne: a dynamic indoor signal map construction and localization system. In *Proceedings of the 4th international conference on mobile systems, applications and services*, 151–164. MobiSys '06, New York, NY, USA: ACM.

[76] Jing, Xiangpeng, S.S. Anandaraman, M.A. Ergin, I. Seskar, and D. Raychaudhuri. Distributed coordination schemes for multi-radio co-existence in dense spectrum environments. In *Ieee dyspan '08*.

[77] K. Lakshminarayanan et. al. Understanding 802.11 performance in heterogeneous environments. HomeNets '11.

[78] K. Sutton and L. E. Doyle. 2007. Cyclostationary signatures for rendezvous in ofdm-based dynamic spectrum access networks.

[79] Kabbani, Abdul. 2006. Distributed low-complexity maximum throughput scheduling in wireless backhaul networks.

[80] Kamerman, A., and L. Monteban. 1997. Wavelan ii: A high-performance wireless lan for the unlicensed band.

[81] Kamerman, A. et al. Microwave oven interference on wireless lans operating in the 2.4 ghz ism band. In *Pimrc'97*.

[82] Kandula, Srikanth, Ranveer Chandra, and Dina Katabi. 2008. What's going on?: learning communication rules in edge networks. In *Proceedings of the acm sigcomm 2008 conference on data communication*, 87–98. SIGCOMM '08, New York, NY, USA: ACM.

[83] Kanodia, V., C. Li, A. Sabharwal, B. Sadeghi, and E. Knightly. 2001. Distributed multi-hop scheduling and medium access with delay and throughput constraints. In *Acm mobicom*.

[84] Kashyap, Anand, Samrat Ganguly, and Samir R. Das. A measurement-based approach to modeling link capacity in 802.11-based wireless networks. In *Mobicom '07*.

[85] Kashyap, Anand, Samrat Ganguly, and Samir R. Das. 2007. A measurement-based approach to modeling link capacity in 802.11-based wireless networks. In *Acm mobicom*.

[86] Katti, Sachin, Shyamnath Gollakota, and Dina Katabi. 2007. Embracing wireless interference: Analog network coding. In *In: Proc. of the acm sigcomm*, 397–408. MIT.

[87] Khan, Malik Ahmad Yar, and Darryl Veitch. 2009. Isolating physical per for smart rate selection in 802.11. In *Infocom*, 1080–1088. IEEE.

[88] Khan, Malik Ahmad Yar, and Darryl Veitch. 2011. Smartrate: A new dynamic rate adaptation algorithm for 802.11 wireless networks. In *Wowmom*, 1–10. IEEE.

[89] Kim, Kyu-Han, and Kang G. Shin. 2006. On accurate measurement of link quality in multi-hop wireless mesh networks. In *Acm mobicom*.

[90] Kriara, Lito, Soï¬'a Pediaditaki, and Mahesh K. Marina. 2012. On the importance of loss differentiation for link adaptation in wireless lans.

[91] Kulkarni, Parag, Benjamin Motz, Tim Lewis, and Sadia Quadri. 2011. Inferring loss causes to improve link rate adaptation in wireless networks. In *Proceedings of the 2011 ieee international conference on advanced information networking and applications*, 659–666. AINA '11, Washington, DC, USA: IEEE Computer Society.

[92] Kullback, S. Information theory and statistics, 1959.

[93] Kumar, Anurag, Eitan Altman, Daniele Miorandi, and Munish Goyal. 2007. New insights from a fixed-point analysis of single cell ieee 802.11 wlans. *IEEE/ACM Transactions on Networking*.

[94] Kumar, Sunil, Vineet S. Raghavan, and Jing Deng. 2006. Medium access control protocols for ad hoc wireless networks: A survey. *Ad Hoc Netw.* 4(3):326–358.

[95] Kumar, V. S. Anil, Madhav V. Marathe, Srinivasan Parthasarathy, and Aravind Srinivasan. 2005. Algorithmic aspects of capacity in wireless networks. In *Sigmetrics*.

[96] Kyriakou, Georgios, Donatos Stavropoulos, Iordanis Koutsopoulos, Thanasis Korakis, and Leandros Tassiulas. 2012. A framework and experimental study for discrimination of collision and channel errors in wireless lans. In *Testbeds and research infrastructure. development of networks and communities*, vol. 90, 271–285. Springer Berlin Heidelberg.

[97] L. Yang, W. Hou, L.Cao, B. Y. Zhao and H. Zheng. Supporting demanding wireless applications with frequency-agile radios. In *Ndsi'08*.

[98] Lakshminarayanan, Kaushik, Samir Sapra, Srinivasan Seshan, and Peter Steenkiste. RFDump: an architecture for monitoring the wireless ether. In *Conext'09*.

[99] Li, Jinyang, Charles Blake, Douglas S.J. De Couto, Hu Imm Lee, and Robert Morris. 2001. Capacity of ad hoc wireless networks. In *Acm mobicom*.

[100] Liang, Chieh-Jan Mike, Nissanka Bodhi Priyantha, Jie Liu, and Andreas Terzis. Surviving Wi-Fi interference in low power zigbee networks. In *Acm sensys'10*.

[101] Like, Eric, Vasu D. Chakravarthy, Paul Ratazzi, and Zhiqiang Wu. 2009. Signal classification in fading channels using cyclic spectral analysis. *EURASIP J. Wirel. Commun. Netw.* 2009:29:1–29:14.

[102] Lin, Kate Ching-Ju, Nate Kushman, and Dina Katabi. 2008. Ziptx: Harnessing partial packets in 802.11 networks. In *Proceedings of the 14th acm international conference on mobile computing and networking*, 351–362. MobiCom '08, New York, NY, USA: ACM.

[103] Liu, Xi, Anmol Sheth, Michael Kaminsky, Konstantina Papagiannaki, Srinivasan Seshan, and Peter Steenkiste. Dirc: increasing indoor wireless capacity using directional antennas. In *Sigcomm '09*.

[104] Mahajan, Ratul, Maya Rodrig, David Wetherall, and John Zahorjan. Analyzing the mac-level behavior of wireless networks in the wild. *SIGCOMM '06*.

[105] Manweiler, Justin, Naveen Santhapuri, Souvik Sen, Romit Roy Choudhury, Srihari Nelakuditi, and Kamesh Munagala. Order matters: Transmission reordering in wireless networks. In *Mobicom'08*.

[106] Mishra, A., E. Rozner, S. Banerjee, and W. Arbaugh. 2005. Exploiting partially overlapping channels in wireless networks: Turning a peril into an advantage. In *Acm/usenix imc*.

[107] Mishra, Arunesh, Vladimir Brik, Suman Banerjee, Aravind Srinivasan, and William A. Arbaugh. 2006. A client-driven approach for channel management in wireless lans. In *Infocom*.

[108] Mishra, Arunesh, Vivek Shrivastava, Dheeraj Agrawal, Suman Banerjee, and Samrat Ganguly. Distributed channel management in uncoordinated wireless environments. In *Mobicom '06*.

[109] Miu, Allen, Hari Balakrishnan, and Can Emre Koksal. 2005. Improving loss resilience with multi-radio diversity in wireless networks. In *Acm mobicom*.

[110] Moscibroda, Thomas, Ranveer Chandra, Yunnan Wu, Sudipta Sengupta, Paramvir Bahl, and Yuan Yuan. Load-aware spectrum distribution in wireless lans. In *Icnp '08*.

[111] MuniWireless. 2011. Smartphones and tablets to drive 350 percent increase in Wi-Fi hotspots by 2015. `http://www.muniwireless.com/2011/11/10/smartphones-tablets-drive-350-percent-increase-wifi-hotspots/`.

[112] Murty, Rohan, Jitendra Padhye, Ranveer Chandra, Alec Wolman, and Brian Zill. 2008. Designing high performance enterprise wi-fi networks. In *Proceedings of the 5th usenix symposium on networked systems design and implementation*, 73–88. NSDI'08, Berkeley, CA, USA: USENIX Association.

[113] Nedevschi, Sergiu, Rabin K. Patra, Sonesh Surana, Sylvia Ratnasamy, Lakshminarayanan Subramanian, and Eric Brewer. 2008. An adaptive, high performance mac for long-distance multihop wireless networks. In *Acm mobicom*.

[114] Niculescu, Dragos. 2007. Interference map for 802.11 networks. In *Imc*.

[115] O. Zakaria. Blind signal detection and identification over the 2.4 GHz ISM band for cognitive radio. In *Ms thesis usf'09*.

[116] O'Shea, T. J., T. C. Clancy, and H. J. Ebeid. 2007. Practical signal detection and classification in GNU Radio. In *Sdr forum technical conference*.

[117] Padhye, J., S. Agarwal, V. Padmanabhan, L. Qiu, A. Rao, and B. Zill. 2005. Estimation of link interference in static multi-hop wireless networks. In *Imc*.

[118] Patra, Rabin, Sergiu Nedevschi, Sonesh Surana, Anmol Sheth, Lakshminarayanan Subramanian, and Eric Brewer. 2007. Wildnet: Design and

implementation of high performance wifi based long distance networks. In *Nsdi*.

[119] Peng, Jun, Liang Cheng, and Biplab Sikdar. 2007. A wireless mac protocol with collision detection. *IEEE Transactions on Mobile Computing* 6(12): 1357–1369.

[120] Phaiboon, Supachai. 2002. An empirically based path loss model for indoor wireless channels in laboratory building. In *Proceedings of ieee tencon*.

[121] Qiu, Lili, Yin Zhang, Feng Wang, Mi Kyung Han, and Ratul Mahajan. 2007. A general model of wireless interference. In *Acm mobicom*.

[122] Quinlan, J. Ross. 1993. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc.

[123] R. Mahajan et. al. Analyzing the mac-level behavior of wireless networks in the wild. *SIGCOMM '06*.

[124] Rahul, Hariharan, Farinaz Edalat, Dina Katabi, and Charles G. Sodini. . Frequency-aware rate adaptation and mac protocols. In *Mobicom '09*.

[125] Rahul, Hariharan, Nate Kushman, Dina Katabi, Charles Sodini, and Farinaz Edalat. . Learning to share: narrowband-friendly wideband networks, SIGCOMM'08.

[126] Ramachandran, Kishore, Ravi Kokku, Honghai Zhang, and Marco Gruteser. Symphony: synchronous two-phase rate and power control in 802.11 wlans. In *Mobisys '08*.

[127] Raman, Bhaskaran, and Kameswari Chebrolu. 2005. Design and evaluation of a new mac protocol for long-distance 802.11 mesh networks. In *Acm mobicom*.

[128] Rao, Ananth, and Ion Stoica. 2005. An overlay mac layer for 802.11 networks. In *Acm mobisys*.

[129] Rappaport, T. 1996. *Wireless communications: Principle and practice*. Prentice Hall.

[130] Rayanchu, Shravan, Arunesh Mishra, Dheeraj Agrawal, Sharad Saha, and Suman Banerjee. 2008. Diagnosing wireless packet losses in 802.11: Separating collision from weak signal. In *Infocom*, 735–743. IEEE.

[131] Rayanchu, Shravan, Ashish Patro, and Suman Banerjee. 2011. Airshark: Detecting non-WiFi devices using commodity WiFi hardware. In *ACM IMC*.

[132] Rayanchu, Shravan, Ashish Patro, and Suman Banerjee. 2012. Catching whales and minnows using WiFiNet: Deconstructing non-WiFi interference using commWiFi hardware. In *ACM USENIX NDSI*.

[133] Reis, C., R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. 2006. Measurement-based models of delivery and interference in static wireless networks. In *Acm sigcomm*.

[134] Research, Infonetics. 2012. It's official: Wireless LAN is the hottest market in enterprise networking. `http://www.infonetics.com/pr/2012/4Q11-Wireless-LAN-and-Ethernet-Switch-Market-Highlights.asp`.

[135] R.Gummadi, D.Wetherall, B.Greenstein and S.Seshan. Understanding and mitigating the impact of rf interference on 802.11 networks. In *Sigcomm '07*.

[136] Rodrig, Maya, Charles Reis, Ratul Mahajan, David Wetherall, and John Zahorjan. 2005. Measurement-based characterization of 802.11 in a hotspot setting. In *Acm sigcomm e-wind*.

[137] Rodrig, Maya, Charles Reis, Ratul Mahajan, David Wetherall, John Zahorjan, and Ed Lazowska. CRAWDAD data set uw/sigcomm2004.

[138] Rozner, Eric, Yogita Mehta, Aditya Akella, and Lili Qiu. 2007. Traffic-aware channel assignment in enterprise wireless lans. In *Icnp*.

[139] S. Gollakota et al. Clearing the RF Smog: Making 802.11 Robust to Cross-Technology Interference. In *Acm sigcomm 2011*.

[140] S. Thongthammachart and H. Olesen. Bluetooth enables in-door mobile location services. *VTC'03*.

[141] Sankaran, Sundar G., Masoud Zargari, Lalitkumar Y. Nathawad, Hirad Samavati, Srenik S. Mehta, Alireza Kheirkhahi, Phoebe Chen, Ke Gong, Babak Vakili-Amini, Justin A. Hwang, Shuo-Wei Mike Chen, Manolis Terrovitis, Brian J. Kaczynski, Sotirios Limotyrakis, Michael P. Mack, Haitao Gan, Meelan Lee, Richard T. Chang, Hakan Dogan, Shahram Abdollahi-Alibeik, Burcin Baytekin, Keith Onodera, Suni Mendis, Andrew Chang, Yashar Rajavi, Steve Hung-Min Jen, David K. Su, and Bruce A. Wooley. 2009. Design and implementation of a cmos 802.11n soc. *Comm. Mag.*

[142] Sen, Souvik, Romit Roy Choudhury, Naveen Santhapuri, and Srihari Nelakuditi. 2009. Moving away from collision avoidance: Towards collision detection in wireless networks. In *Hotnets'09*.

[143] Sen, Souvik, Romit Roy Choudhury, and Srihari Nelakuditi. 2010. Csma/cn: carrier sense multiple access with collision notification. In *Proceedings of the sixteenth annual international conference on mobile computing and networking*, 25–36. MobiCom '10, New York, NY, USA: ACM.

[144] Sen, Souvik, Naveen Santhapuri, Romit Roy Choudhury, and Srihari Nelakuditi. 2010. Accurate: constellation based rate estimation in wireless networks. In *Proceedings of the 7th usenix conference on networked systems design and implementation*, 12–12. NSDI'10, Berkeley, CA, USA: USENIX Association.

[145] Sheth, Anmol, Christian Doerr, Dirk Grunwald, Richard Han, and Douglas C. Sicker. 2006. Mojo: a distributed physical layer anomaly detection system for 802.11 wlans. In *Mobisys*.

[146] Shin, Soo Young, Jeong Seok Kang, and Hong Seong Park. Packet error rate analysis of zigbee under wlan and bluetooth interferences. In *Ieee trans. wireless comm.'06*.

[147] Shrivastava, S., S. Rayanchu, J. Yoon, and S. Banerjee. 2008. 802.11n under the microscope. In *Imc*.

[148] Shrivastava, Vivek, Nabeel Ahmed, Shravan Rayanchu, Suman Banerjee, Srinivasan Keshav, Konstantina Papagiannaki, and Arunesh Mishra. CENTAUR: realizing the full potential of centralized wlans through a hybrid data path. In *Mobicom'09*.

[149] Shrivastava, Vivek, Shravan Rayanchu, Suman Banerjee, and Konstantina Papagiannaki. 2011. Pie in the sky: online passive interference estimation for enterprise wlans. In *Proceedings of the 8th usenix conference on networked systems design and implementation*, 25–25. NSDI'11, Berkeley, CA, USA: USENIX Association.

[150] Taher, M. et al. Characterization of an unintentional wi-fi interference device-the residential microwave oven. In *Milcom'06*.

[151] Tan, Godfrey, and John Guttag. 2004. Time-based fairness improves performance in multi-ra te wlans. In *Proc. of usenix*.

[152] Tan, Kun, Ji Fang, Yuanyang Zhang, Shouyuan Chen, Lixin Shi, Jiansong Zhang, and Yongguang Zhang. Fine-grained channel access in wireless lan, SIGCOMM'10.

[153] Tech, Mashable. 2011. 5 Reasons YouâŁ™re Consuming More Mobile Content. `http://mashable.com/2011/04/28/mobile-content/`.

[154] TechCrunch. 2011. 61 Percent of U.S. Households Now Have WiFi. `http://techcrunch.com/2012/04/05/study-61-of-u-s-households-now-have-wifi/`.

[155] Vaidya, N., P. Bahl, and S. Gupta. 2000. Distributed fair scheduling in a wireless lan. In *Acm mobicom*.

[156] Vutukuru, M., H. Balakrishnan, and K. Jamieson. 2009. Cross-layer wireless bit rate adaptation. In *Acm sigcomm*.

[157] Vutukuru, Mythili, Hari Balakrishnan, and Kyle Jamieson. 2009. Cross-Layer Wireless Bit Rate Adaptation. In *Acm sigcomm*. Barcelona, Spain.

[158] Vutukuru, Mythili, Kyle Jamieson, and Hari Balakrishnan. Harnessing Exposed Terminals in Wireless Networks. In *Nsdi'08*.

[159] W. Gardner. Exploitation of spectral redundancy in cyclostationary signals. In *Ieee signal processing magazine'91*.

[160] Whitehouse, K., A. Woo, F. Jiang, J. Polastre, and D Culler. 2005. Exploiting the capture effect for collision detection and recovery. In *Emnets-11*.

[161] Wireless, Daily. 2011. Top Wireless Trends of 2011. `http://www.dailywireless.org/2011/12/29/top-wireless-trends-of-2011/`.

[162] Wong, S., S. Lu, H. Yang, and V. Bhargavan. 2006. Robust rate adaptation for 802.11 wireless networks. In *Acm mobicom*.

[163] Wong, S., H. Yang, S. Lu, and V. Bhargavan. 2006. Robust rate adaptation for 802.11 wireless networks. In *Acm mobicom*.

[164] Yang, Lei, Lili Cao, Heather Zheng, and Elizabeth Belding. Traffic-aware dynamic spectrum access. In *Wicon '08*.

[165] Yoo S-H, Choi J-H, Hwang J-H, and Yoo C. 2005. Eliminating the performance anomaly of 802.11b. In *In lecture notes in computer science*.

[166] Youssef, Moustafa, and Ashok Agrawala. The horus wlan location determination system. In *Mobisys'05*.

[167] Yu, Guoshen, StÃ©phane Mallat, and Emmanuel Bacry. 2008. Audio denoising by time-frequency block thresholding. *IEEE Transactions on Signal Processing* 56:1830–1839.

[168] Yuan, Yuan, Paramvir Bahl, Ranveer Chandra, Thomas Moscibroda, and Yunnan Wu. Allocating dynamic time-spectrum blocks in cognitive radio networks. In *Mobihoc '07*.

[169] Yuan, Y. et al. Knows : Kognitiv networking over white spaces. *IEEE DySPAN'07*.

[170] Zeng, Zheng, and P. R. Kumar. 2008. Towards optimally exploiting physical layer information in ofdm wireless networks. In *Proceedings of the 4th annual international conference on wireless internet*, 33:1–33:9. WICON '08, ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).