T-SIMn: Towards a Framework for the Trace-Based Simulation of 802.11n Networks

by

Andrew Heard

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2016

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

With billions of WiFi devices now in use, and growing, combined with the rising popularity of high-bandwidth applications, such as streaming video, demands on WiFi networks continue to rise. To increase performance for end users the 802.11n WiFi standard introduces several new features that increase Physical Layer Data Rates (PLDRs). However, the rates are less robust (i.e., more prone error). Optimizing throughput in an 802.11n network requires choosing the combination of features that results in the greatest balance between PLDRs and error rates, which is highly dependent on the environmental conditions. While the faster PLDRs are an important factor in the throughput gains afforded by 802.11n, it is only when they are used in combination with the new MAC layer features, namely Frame Aggregation (FA) and Block Acknowledgments (BAs), that 802.11n achieves significant gains when compared to the older 802.11g standard. FA allows multiple frames to be combined into a large frame so that they can be transmitted and acknowledged as one aggregated packet, which results in the channel being used more efficiently.

Unfortunately, it is challenging to experimentally evaluate and compare the performance of WiFi networks using different combinations of 802.11n features. WiFi networks operate in 2.4 and 5 GHz bands, which are shared by WiFi devices, included in computers, cell phones and tablets; as well as Bluetooth devices, wireless keyboards/mice, cordless phones, microwave ovens and many others. Competition for the shared medium can negatively impact throughput by increasing transmission delays or error rates. This makes it difficult to perform repeatable experiments that are representative of the conditions in which WiFi devices are typically used. Therefore, we need new methodologies for understanding and evaluating how to best use these new 802.11n features.

An existing trace-based simulation framework, called T-RATE, has been shown to be an accurate alternative to experimentally evaluating throughput in 802.11g networks. We propose T-SIMn, an extension of the T-RATE framework that includes support for the newer 802.11n WiFi standard. In particular, we implement a new 802.11n network simulator, which we call SIMn. Furthermore, we develop a new implementation of the trace collection phase that incorporates FA. We demonstrate that SIMn accurately simulates throughput for one, two and three-antenna PLDRs in 802.11n with FA. We also show that SIMn accurately simulates delay due to WiFi and non-WiFi interference, as well as error due to path loss in mobile scenarios. Finally, we evaluate the T-SIMn framework (including trace collection) by collecting traces using an iPhone. The iPhone is representative of a wide variety of one antenna devices. We find that our framework can be used to accurately simulate these scenarios and we demonstrate the fidelity of SIMn by uncovering problems with our initial evaluation methodology. We expect that the T-SIMn framework will be

suitable for easily and fairly evaluating rate adaptation, frame aggregation and channel bandwidth adaptation algorithms for 802.11n networks, which are challenging to evaluate experimentally.

Acknowledgements

I would first like to thank my supervisor, Professor Tim Brecht, for his guidance, support and understanding throughout this degree. I am grateful for his encouragement and optimism that enabled me to persevere during the most challenging portions of this project. I would also like to thank my committee members, Professor Martin Karsten and Professor Srinivasan Keshav, for their valuable suggestions that improved this thesis.

I would like to thank my colleague, Ali Abedi, for his tremendous help and friendship over the last two years. From debugging kernel panics to assembling furniture, Ali was always happy to help and I couldn't have asked for a better teammate.

I appreciate the generous financial support provided by the Natural Sciences and Engineering Research Council, the Government of Ontario and the University of Waterloo.

Last but not least, I dedicate this thesis to my parents, Sylvia and Stephen Heard, who have given me love and support my entire life. I can't imagine getting here without you.

Table of Contents

List of Tables

List of Figures

List of Acronyms

ADC Analog-to-Digital Converter. 32, 39

AP Access Point. 9, 20, 21, 26, 29, 32, 35, 39, 40, 42–44, 46, 50, 54, 59

BA Block Acknowledgment. iii, 14–16, 20, 33, 34, 37, 65

BAR Block-Ack Request. 34

BAW Block-Ack Window. 34, 35, 49, 57, 58

CB Channel Bandwidth. 2, 14, 16–18, 22, 28, 30, 33, 39, 40, 46, 54, 59, 62, 64, 65, 67

CCI Cycle Counter Information. 9–11, 19, 32, 39

CSI Channel State Information. 64

CSMA/CA Carrier Sense Multiple Access with Collision Avoidance. 38, 50

DIFS Distributed Inter-Frame Space. 32–34, 38

FA Frame Aggregation. iii, 2–4, 14–16, 19, 33–37, 44, 53, 57, 60, 61, 63, 65

FAA Frame Aggregation Algorithm. 3, 22, 23, 62, 65

FHSS Frequency-Hopping Spread Spectrum. 45, 46

FPGA Field-Programmable Gate Array. 7

GI Guard Interval. 2, 14, 16–18, 22, 30, 32, 33, 40, 54, 64, 67

HT-LTF High-Throughput Long Training Field. 30

JIT Just In Time. 21

LGI Long Guard Interval. 14, 18, 28, 32, 62

MBP MacBook Pro. 50, 51

MCS Modulation and Coding Scheme. 2, 3, 8, 10, 14, 16–18, 20, 30, 39, 54, 57, 64

MIMO Multiple-Input and Multiple-Output. 30

PLCP Physical Layer Convergence Protocol. 30

PLDR Physical Layer Data Rate. iii, 1–3, 13–18, 30–35, 41, 48, 67

RAA Rate Adaptation Algorithm. 3, 6, 9, 11, 12, 16, 22, 23, 53, 54, 62–65

RSSI Received Signal Strength Indicator. 20

RTS Request-to-Send. 20

SFIER Subframe Index Error Rate. 21–23, 28, 49, 50, 52, 65

SGI Short Guard Interval. 3, 14, 18, 20, 28, 29, 31, 32, 62

SIFS Short Inter-Frame Space. 32–34

SNR Signal-to-Noise Ratio. 7, 8

SS Spatial Stream. 2, 14, 16–18, 28–31, 54, 59, 61, 64, 65, 68

TWTT Third-party WiFi Traffic Trace. 10–12

UDP User Datagram Protocol. 21, 35

Chapter 1

Introduction

Thesis Statement:

> We believe that it is possible to build a framework for the trace-based simulation of 802.11n devices with one antenna.

This thesis describes the design, implementation and evaluation of T-SIMn, a framework for the trace-based simulation of 802.11n devices. T-SIMn is based on an existing trace-based simulation framework for 802.11g devices, called T-RATE. T-SIMn consists of a trace collection methodology and a trace-driven 802.11n network simulator that we call SIMn. Although the focus of this thesis is the simulator (SIMn), we also demonstrate that the T-SIMn framework is highly accurate in the simulation of 802.11n devices with one antenna.

1.1 Motivation

With billions of WiFi devices now in use, and growing, combined with the rising popularity of high-bandwidth applications, such as streaming video, demands on WiFi networks continue to rise. Since its first release in 1997, the 802.11 WiFi standard has seen many amendments to increase performance for end users. 802.11 WiFi networks support multiple Physical Layer Data Rates (PLDRs) rates to handle a range of wireless channel conditions. Generally, slower PLDRs are more robust (i.e., are less prone to error) but offer lower potential throughput, whereas faster PLDRs are more fragile (i.e., are more prone to error) but offer higher potential throughput. The 802.11g standard supports 8 different PLDRs

through a combination of Modulation and Coding Schemes (MCSs). The newer 802.11n standard introduces several new physical layer features (Multiple Spatial Streams (SSs) using one to four antennas, Short Guard Intervals (GIs) and 40 MHz Channel Bandwidths (CBs)) to increase throughput. We refer to the combination of features and an MCS as a rate configuration. Combinations of these features and MCSs results in up to 128 different rate configurations (i.e., 8 $MCSs \times 4\ SSs \times 2\ GIs \times 2\ CBs = 128$). In order to optimize throughput in an 802.11n network, we must choose the rate configuration that results in the greatest balance between PLDRs and error rates, which is highly dependent on the environmental conditions. However, it is challenging to experimentally evaluate and compare the performance of WiFi networks using different rate configurations.

WiFi networks operate in 2.4 and 5 GHz bands of the radio spectrum, which are license-free in most countries [12]. These bands are shared by WiFi devices, included in computers, cell phones and tablets; as well as Bluetooth devices, wireless keyboards/mice, cordless phones, microwave ovens and many others. If one or more of these devices are competing with a WiFi device for access to the channel, they can negatively impact throughput by increasing transmission delays or error rates. This makes it difficult to perform repeatable experiments that are representative of the conditions in which WiFi devices are typically used. Therefore, we need new techniques for understanding and evaluating how to best use these new 802.11n features.

Abedi and Brecht [2] propose a solution that uses traces that capture environmental conditions, rather than models, to simulate 802.11g networks. This framework is called T-RATE and forms the basis of this thesis. This thesis focuses on extending T-RATE to more widely used 802.11n networks, using a framework we call T-SIMn. In particular, we focus on the simulator component of the framework, called SIMn. We expected that this would be a relatively simple extension of T-RATE (i.e., updating the existing implementation of T-RATE to support the many new, and faster, Physical Layer Data Rates (PLDRs) in 802.11n). However, extending T-RATE to support 802.11n turned out to be a much more interesting problem than anticipated. While the faster PLDRs are an important factor in the throughput gains afforded by 802.11n, it is only when they are used in combination with MAC layer changes, namely Frame Aggregation (FA), that 802.11n achieves its significant increases in throughput when compared with 802.11g. FA allows multiple frames to be combined into a large frame so that they can be transmitted and acknowledged as one aggregated packet, which results in the channel being used more efficiently.

To demonstrate the importance of FA in obtaining high throughput, Figure 1.1 shows the maximum theoretical throughput obtained using the highest PLDRs in 802.11n for one, two and three antennas, respectively (when aggregating 1, 2, 4, 8, 16 or 32 frames, with "1" meaning that frame aggregation is not being used). Without frame aggregation, we see

limited gains in throughput when comparing the one antenna (150 Mbps) PLDR, the two antenna (300 Mbps) PLDR and the three antenna (450 Mbps) PLDR, despite the threefold increase in PLDR when going from 1 to 3 antennas. However, when aggregating up to 32 frames, we see a nearly threefold increase in throughput between the 1 and 3 antenna configurations. Because performance is so heavily dependent on FA, accurate simulation of FA is crucial for T-SIMn to be useful in the study of a range of active research topics. This includes the evaluation of Frame Aggregation Algorithms (FAAs) [8, 36], Rate Adaptation Algorithms (RAAs) [11, 38, 29, 39] and Channel Bandwidth Adaptation [10, 16]. As well as 802.11n Link Adaptation [21], which studies the above topics together. For this reason, simulation of FA in environments that are representative of those in which WiFi devices are used is the focus of the T-SIMn framework.

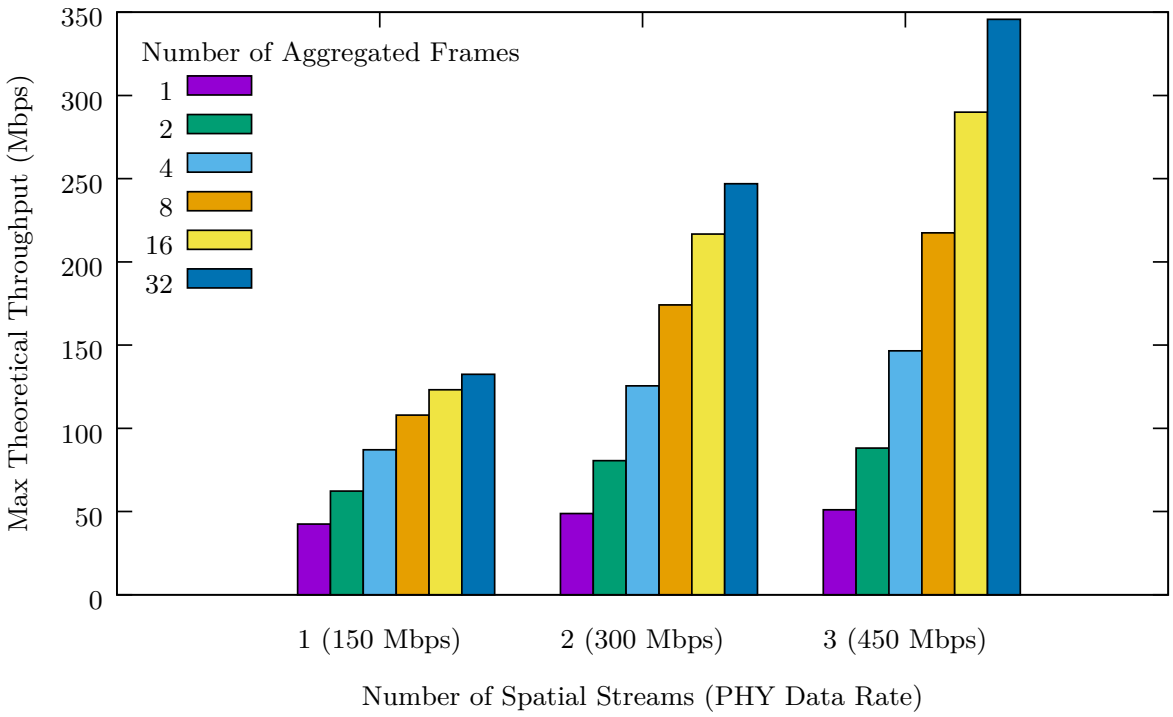

Figure 1.1: The maximum achievable throughput using the fastest 802.11n rate configuration (Modulation and Coding Scheme (MCS) 7, Short Guard Interval (SGI), 40 MHz) for 1, 2 and 3 spatial streams (i.e., antennas) when aggregating between 1 and 32 frames.

1.2 Contributions

The major contribution of this thesis is to extend the T-RATE framework to 802.11n networks, which we call T-SIMn. Figure 1.2 provides an overview of the pre-existing T-RATE framework. While the overall framework remains unchanged from T-RATE, we make significant implementation changes in each phase to support 802.11n networks.

The contributions of this thesis are:

- We implement 802.11n trace collection in the Ath9k driver. We capture the state of Frame Aggregation (FA) by logging each aggregated frame, including the fate of each subframe in a bitmap. We also capture the rate configuration (i.e., which 802.11n physical layer features are used) for each aggregated frame.
- We eliminate trace preparation as a separate phase and no longer generate "complete traces" because, when compared with 802.11g, this is time consuming and requires a considerable amount of disk space, due to the many rate configurations in 802.11n. We implement the conceptual functionality of trace preparation within the trace processing (simulation) phase, computing frame fates only as needed.
- We demonstrate the need to compute Subframe Index Error Rates (SFIERs) to accurately simulate frame fates with FA.
- We implement a trace-driven 802.11n network simulator, SIMn, including an FA Algorithm that mimics the behavior of the Ath9k driver.
- We evaluate the accuracy of the SIMn simulator with FA for one, two and three-antenna rates in 802.11n, with both WiFi and non-WiFi interference; and errors due to path loss.
- We evaluate the accuracy of the T-SIMn framework with trace collection for all one-antenna rates without RTS/CTS, in a mobile scenario.
- We describe how we use the simulator to discover a flaw with our initial methodology for evaluating the accuracy of the T-SIMn framework. After correcting the flaw in our methodology, we show that the framework is highly accurate.

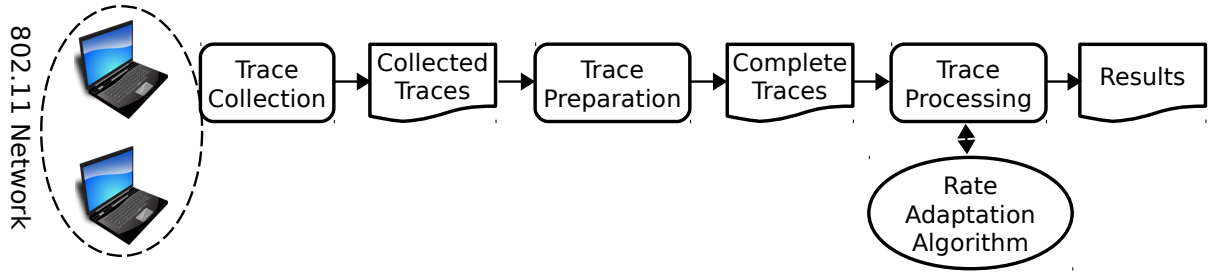

Figure 1.2: Overview of the T-RATE Framework.

1.3 Thesis Organization

We begin by describing T-RATE, the framework that T-SIMn is based on, and providing an overview of the new features of 802.11n, in Chapter 2. We then describe the implementation of the T-SIMn framework, in Chapter 3. We describe and evaluate the simulation (with SIMn) of the new 802.11n features, in Chapter 4; simulation of channel access due to WiFi and non-WiFi interference, in Chapter 5; and simulation of channel error rates due to path loss, in Chapter 6. In Chapter 7, we evaluate the use of T-SIMn for widely used one-antenna devices, as an example of how the framework can be used, in an environment that is representative of that in which WiFi devices are typically used. Lastly, we discuss the limitations of T-SIMn, propose future areas of research and provide concluding remarks in Chapter 8.

Chapter 2

Background and Related Work

T-SIMn builds upon an existing project called T-RATE [2]. T-RATE is a trace-driven framework for evaluating Rate Adaptation Algorithms (RAAs) designed for 802.11g networks [2]. While the evaluation of RAAs is not the focus of this thesis, T-RATE serves as an excellent base for T-SIMn because of its trace collection methodology, as well as its highly accurate simulation of 802.11g networks. T-SIMn adapts these approaches to the newer 802.11n standard, which introduces many new features at the physical and MAC layers. We begin by discussing alternatives to trace-based simulation in Section 2.1. We then describe the existing implementation of T-RATE in Section 2.2. Lastly, we discuss 802.11n and the new features that it adds in Section 2.3.

2.1 Alternative Approaches to Performance Evaluation

We begin by describing different methodologies for evaluating the performance of 802.11 networks. We argue that if trace-based simulations can be implemented for 802.11n networks, they provide much promise for the fair evaluation and comparison of algorithms that are critical for obtaining high throughput in 802.11n networks.

2.1.1 Experiments

Conducting experiments is a common technique for evaluating the performance of WiFi networks. The advantage of this approach is that real-world wireless channel conditions are captured. However, it is challenging to conduct repeatable experiments because channel

conditions can vary significantly between trials due to many factors, including mobility, changes in the environment (including the movement of people nearby), and the presence of WiFi and non-WiFi interferers [7]. To reduce variation in channel conditions, experiments are often conducted in environments that are not representative of those in which WiFi devices are typical used, such as "in the middle of the night" or "when no one else is around" [28, 9]. In previous work, we propose a technique called *multiple interleaved trials* [3] that allows WiFi performance to be fairly and repeatably compared in challenging scenarios with mobility and interference. However, this technique is highly time consuming, especially as the number of competing alternatives becomes large. Moreover, it relies on being able to quickly and easily switch between alternatives that are being compared, which is not always possible.

2.1.2 Emulation

Emulation is another alternative for evaluating and studying 802.11 networks. Although emulation is a broad term, in the literature it commonly refers to a system that uses real WiFi devices connected by wire to a Field-Programmable Gate Array (FPGA) which simulates signal propagation [17]. An emulation testbed by Judd and Steenkiste [18, 20, 19] is one of the most prominent examples of this type of system. As real devices are used, the MAC and higher layers are *not* simulated. However, the physical layer *is simulated* using the FPGA to alter the signals being transmitted between devices. The major disadvantage of this approach is that the realism of the emulation is limited by the models used to simulate the physical layer. Since the MAC layer uses deterministic highly specified protocols, we can more easily simulate the MAC layer than the physical layer. As a result, our approach is to collect traces that capture the physical layer impact on frame fates and to simulate the MAC layer. Hybrid approaches using traces to simulate the physical layer and emulation for the MAC layer have also been proposed [18, 41, 40], however these have been limited to 802.11b networks and rely on measurements of Signal-to-Noise Ratio (SNR) or signal strength to simulate the physical layer. However, SNR and signal strength are shown to be inaccurate predictors of frame fates [9, 13], especially in 802.11n networks [3].

2.1.3 Simulation

Simulation is a common technique for evaluating the performance of 802.11 networks. Since both the physical and MAC layers are simulated in this approach, comparisons are repeatable. However, they may not reflect real-world performance due to the challenging

nature of simulating wireless signals in the physical environment. WiFi signals are impacted by many factors, including the distance between devices, material types of surrounding objects and walls, wavelength, mobility and many more [34]. *ns-3* [27] is a popular network simulator that is still under active development. The physical layer simulation in *ns-3* consists of three major components [22], a signal propagation model, an error rate model and an interference model. The propagation model simulates the transmission of a wireless signal from a sender and estimates the SNR at the receiver [34], (based on the distance between devices, material types of surrounding objects and walls, wavelength, mobility and many more factors). The error rate model estimates the frame fate based on the SNR, the Modulation and Coding Scheme (MCS) [15] and based on energy received on the channel as interference [27]. Each of these models have multiple implementations, each of which makes trade-offs between computational complexity and accuracy [34]. Other popular 802.11 network simulators include OPNET Modeler [35] (recently renamed to Riverbed Modeler) and QualNet [30]. The complexity and time varying nature of all of the factors that can affect a frame's fate makes it incredibly challenging to obtain accurate results, especially since models of one environment (e.g., one home) may not necessarily work in another environment (e.g., an office or even a different home) and in the case of environments with mobility, this model may be changing over time. Rather than simulating the physical layer, we collect traces to capture the impact of the physical layer on frame fates, allowing us to handle more complex scenarios than can be accurately modeled by traditional simulators.

2.1.4 Summary

Experiments, emulation and simulation are all viable alternatives to the trace-based simulator that we propose in this thesis. Each technique provides its own advantages and disadvantages. At one end of the spectrum, experiments provide complete realism but repeatability is challenging. On the other, simulations provide repeatability but realistically simulating the physical layer is challenging. We propose the use of trace-based simulation to avoid simulating the effect that the highly volatile, and environmentally sensitive, physical layer can have on frame fates. Instead, we rely on traces to capture the impact of the physical layer on frame fates. We simulate the precisely defined MAC layer since it can be simulated with a high degree of accuracy because it is not affected by the physical environment. We believe that this provides an excellent combination of repeatability and accuracy.

2.2 T-RATE (802.11g)

T-RATE is a trace-driven framework for evaluating RAAs designed for 802.11g [2]. T-RATE eschews modelling or emulation of wireless channel conditions in favour of traces that capture channel access and channel error rate information. These traces are used to simulate RAAs in conditions that are more representative of those in which 802.11 WiFi is typically used (i.e., in the presence of interferers, both WiFi and non-WiFi, and path loss). Since a single trace is used to evaluate multiple competing RAAs, fair comparisons can be made in a wide variety of scenarios, as each RAA is exposed to the exact same channel conditions.

T-RATE employs a three phase approach to simulation. First, a trace is collected by saturating a wireless network while cycling between all available 802.11g transmission rates; the fate of these frames, the delays they experience and received third-party WiFi frames are recorded to form the collected trace. Next, for each 802.11g transmission rate, the trace preparation phase estimates packet fates for the points in time that fall between collected samples to produce a complete trace. Lastly, the trace processing phase simulates an RAA by using the frame fates and delays obtained from the complete traces. We now discuss each of the three phases of T-RATE in detail.

2.2.1 Trace Collection

The purpose of the trace collection phase in T-RATE is to capture channel access and channel error rate information (i.e., a trace), for each 802.11g transmission rate. This trace can later be used to simulate the behavior of one or more RAAs using conditions that very closely approximate those in which the trace was collected. In the trace collection phase, an 802.11g Access Point (AP) is created on a computer using hostapd [24]. A network is established between the computer running the AP and another computer, with one acting as a *sender* and the other as a *receiver*. In the evaluation of T-RATE, the AP is always the *receiver* for consistency, though this is not required by the framework. The *sender*'s WiFi card uses a modified version of the Ath9k driver, which logs the Cycle Counter Information (CCI) that measures the number of clock cycles the device spent transmitting and receiving, as well as the number of cycles that the channel was busy. As described in Section 3.3.2, CCI can be used for determining non-WiFi delay. The *sender* runs tcpdump, which captures all outgoing frames and their fate (i.e., whether the frame is acked), as well as all incoming frames, both from the *receiver* (such as Acks) and from third-party WiFi devices. To collect a trace, the *sender* saturates the link to the *receiver*

by sending as many packets as possible using Iperf [37]. During collection, the *sender* cycles through each of the eight 802.11g rates (shown in Appendix A), in a round-robin fashion. After each transmitted frame, the *sender* switches to the next available rate; wrapping back around after all rates have been sampled. This round-robin sampling of rates continues for the entire duration of the trace collection. After the desired amount of time has elapsed, the CCI logs produced by the modified Ath9k driver (example shown in Figure 2.1); the *sender*-side tcpdump log (example shown in Figure 2.2) and the Third-party WiFi Traffic Trace (TWTT) frames (example shown in Figure 2.3) are stored. These three logs constitute the trace that is passed on to the trace preparation phase.

```
[61164.046938] [TRACE] 23971 2151 28370 33835
```

Figure 2.1: Example CCI log line for a frame in T-RATE. Read from left to right, this means the frame was sent at kernel time 61164.046938 (in seconds since system boot time), it has the tag "TRACE", it took 23,971 clock cycles to transmit the data, 2,151 cycles to receive the Ack, and for 28,370 cycles, the card was busy (transmitting, receiving or waiting for channel access). The total time spent on the frame (busy and idle) was 33,835 cycles.

```
33.509061000 40 24 2456
```

Figure 2.2: Example log line for a data frame in T-RATE. Read from left to right, this means the frame was sent at time 33.509061000 seconds, the entry type is type 40 (a data frame), it was transmitted at 24 Mbps (MCS 4), and has the sequence number 2,456.

```
46.746295000 124 26 6
```

Figure 2.3: Example log line for a TWTT frame in T-RATE. Read from left to right, this means the frame was sent at time 46.746295000 seconds, it is 124 bytes including headers, the radiotap header is 26 bytes and it was transmitted at 6 Mbps (MCS 0).

2.2.2 Trace Preparation

At a conceptual level, the goal of the trace preparation phase is to produce a set of lookup tables that can later be used by the trace processing engine to determine, for a time t and a rate r, both the fate of a frame (i.e., acked or not acked) and the transmission delay due to WiFi and non-WiFi interference. In practice, these lookup tables are stores as log files, one for each rate r, which are referred to as a complete traces. Additionally, there is another log file for storing WiFi traffic. It is important to note that for any given time t, the collected trace contains a sample for a single rate, as the *sender* can only sample one rate at time. In contrast, the complete traces include the expected fate of the frame and the packet delay for all rates, at all times t. Since time is a continuous measure with infinitely many points, frame fates and packet delays cannot truly be stored for all times t. Instead, T-RATE discretizes time using a stepping interval on the order of microseconds and stores frame fates and packet delays at these discrete points in time. To fill in the missing values between samples of a rate r in the collected trace, T-RATE relies on the concept of channel coherence time; the duration of time over which channel conditions can be considered constant [2]. A channel coherence time of 100 ms is used by T-RATE, with an averaging window of $t - 50$ ms to $t + 50$ ms (i.e., the 50 ms before and after the given time t). We begin by discussing the methodology used by T-RATE for computing channel error rate (frame fates), and then follow with the methodology for channel access (delay).

To compute the frame fate at time t for rate r, T-RATE considers all samples for the rate r, in the averaging window ($t - 50$ ms to $t + 50$ ms), from the collected trace. The number of these samples that have failed (i.e., frames that are not Acked) is divided by the total number of these samples, to produce an error rate e. A uniform random number is then drawn, in the range [0,1.0]. A frame at time t for rate r is considered successful if the random number is larger than the error rate e. This process is repeated for all time steps t and rates r and is referred to as completing the trace.

To compute the non-WiFi delay at time t for rate r, T-RATE again considers all samples from the collected trace for the rate r, in the averaging window ($t-50$ ms to $t+50$ ms). The delay for a sample is computed by comparing the expected transmission time for a frame sent at rate r against the actual transmission time measured using the CCI. The delays are simply averaged and stored alongside the frame fates in the complete trace. WiFi delays (due to third-party WiFi traffic) are treated differently. Inbound frames, other than Acks, are considered third-party traffic and are logged in a separate file, called the TWTT. The complete traces and the TWTT are then passed to the trace processing phase. Since these traces are RAA-agnostic, they may be re-used for many simulations.

2.2.3 Trace Processing

The trace processing phase uses the complete traces to simulate an RAA. The T-RATE simulator performs a time based simulation of an 802.11g scenario, where the *sender* saturates the link to the *receiver* by sending as many packets as possible. Starting at time $t = 0$ seconds, the simulation proceeds using an event loop that performs the following steps:

1. The TWTT is checked for frames that were received before t. If any such frame exist, t is incremented by the duration of these frames (i.e., the delay incurred by these frames). This is repeated until all frames received before t have been processed.
2. A transmission rate r is obtained from a Rate Adaptation Algorithm (RAA).
3. The delay due to non-WiFi interference is obtained from the complete trace for rate r, which advances t by the duration of the delay.
4. The fate of a frame transmitted at time t, using rate r, is obtained from the complete trace for r. Successful frames increment a counter, which is used to report throughput at the desired simulation time interval. The frame fate is also reported to the RAA in order to drive subsequent rate selection decisions.

This process repeats until the simulation ends; this is generally after the elapsed simulation time is equal to the duration of the collected trace. Figure 2.4 provides an overview of this process.

2.2.4 Summary

T-RATE is a three phase, trace-driven framework for evaluating Rate Adaptation Algorithms (RAAs) and simulating 802.11g networks. T-RATE begins with collecting a trace that represents channel conditions during an experiment. It then pre-computes frame fates and delays for discretized time t, for the duration of the collected trace. Lastly, T-RATE simulates a scenario, where a link between a *sender* and *receiver* is saturated. Overall, these three phases allow a scenario to be simulated that mimics the channel conditions that occurred during trace collection. However, T-RATE is limited to 802.11g networks and does not support the new features of widely used 802.11n networks, which are described in the following section.

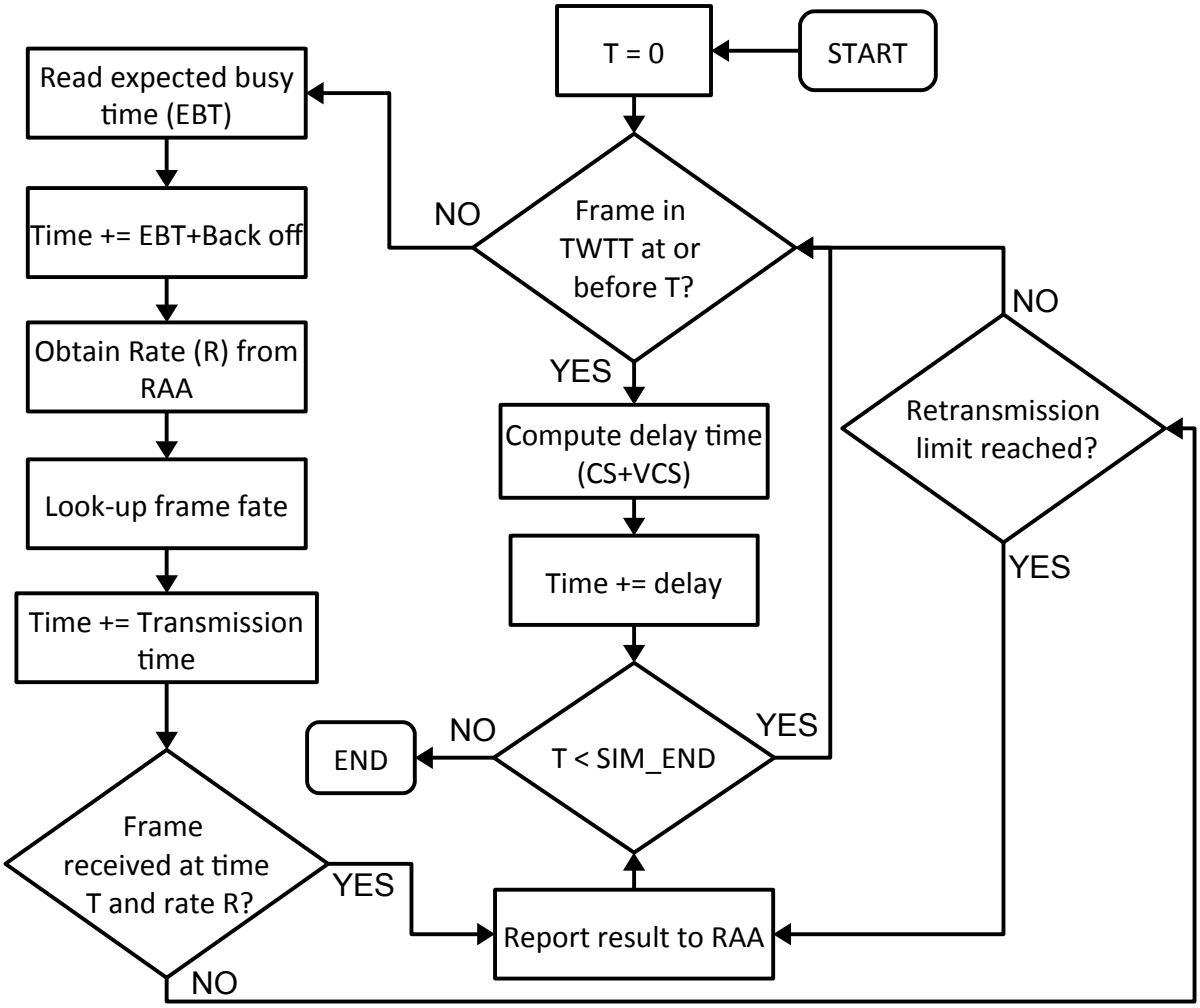

Figure 2.4: T-RATE simulation flowchart [2].

2.3 Overview of 802.11n

802.11n is the "High-Throughput" (HT) successor to the popular 802.11g WiFi standard, which makes improvements at both the physical and MAC layers. In order to fully appreciate the throughput improvements afforded by 802.11n, we now stress the important difference between a Physical Layer Data Rate (PLDR) and realized throughput. 802.11n devices are often advertised as supporting up to 150, 300 or 450 Mbps, with one, two and

three antennas, respectively. These values are referring to maximum PLDRs and these devices do, in fact, transmit data at up to 450 Mbps. However, these devices will never achieve throughput of 450 Mbps due to the overhead of the 802.11n protocol. Since WiFi uses a shared medium for communication, devices must listen and wait for their turn to transmit, in order to avoid interfering with one another; they must also wait for acknowledgments to verify that each packet is received correctly. As the PLDRs become higher and higher, these wait times begin to dominate and decrease the realized throughput. We now examine the physical and MAC layer features that are introduced in the 802.11n standard.

2.3.1 Physical Layer Features

802.11n introduces several new features that increase PLDRs, including multiple Spatial Streams (SSs), which allow multiple data streams to be transmitted simultaneously using different antennas; Short Guard Intervals (SGIs), which reduces the time between transmission of successive symbols in a data stream, from 800 ns to 400 ns; and Channel Bonding, which allows two adjacent 20 MHz channels to be bonded together, providing a Channel Bandwidth (CB) of 40 MHz. These features are described and analyzed in greater detail in Section 4.1. 802.11n also adjusts the available Modulation and Coding Schemes (MCSs), including the addition of a higher coding rate (i.e., less redundancy). An overview of the physical layer features in 802.11g and 802.11n is shown in Table 2.1. For comparison, tables showing all rates for 802.11g and 802.11n can be found in Appendix A.

Table 2.1: Physical Layer Features in 802.11g and 802.11n

Feature	802.11g	802.11n
Modulation and Coding Schemes	8	8
Spatial Streams	1	Up to 4
Guard Intervals	800 ns	800 ns (LGI) and 400 ns (SGI)
Channel Bandwidths	20 MHz	20 MHz and 40 MHz
Number of Rate Configurations	8	Up to 128

2.3.2 MAC Layer Features

802.11n introduces two MAC layer features that increase the efficiency of the 802.11 protocol, including Block Acknowledgments (BAs), which allow multiple packets to be acknowledged at once; and Frame Aggregation (FA), which allows multiple frames to be combined

into a large frame so that they can be transmitted and acknowledged (with a BA) as one aggregated packet, as shown in Figure 2.6. This means that the device only needs to wait for its turn to transmit once for the entire aggregated frame, rather than for each individual frame as shown in Figure 2.5. Together, BAs and FA increase the efficiency of the 802.11 protocol by reducing the amount of time a device spends waiting for its turn to transmit. BAs and FA are described and analyzed in greater detail in Section 4.2.

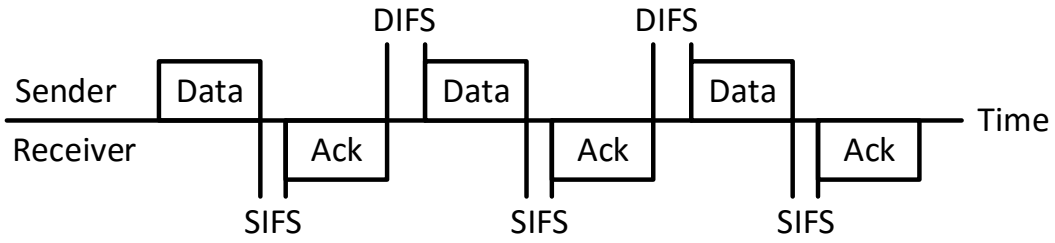

Figure 2.5: Transmission of individual frames without frame aggregation (reproduction from Figure 5-7 in 802.11n: A Survival Guide, by Matthew S. Gast [12]).

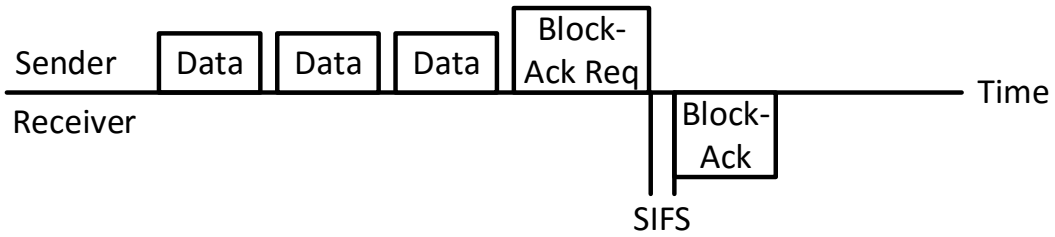

Figure 2.6: Transmission of multiple frames using frame aggregation (reproduction from Figure 5-7 in 802.11n: A Survival Guide, by Matthew S. Gast [12]).

2.3.3 Summary

802.11n increases the realized throughput of WiFi networks using two very different approaches, first by increasing the available Physical Layer Data Rates (PLDRs) and then by improving the efficiency of the protocol with Block Acknowledgments (BAs) and Frame Aggregation (FA). Recall from Figure 1.1 that without FA (i.e., # Aggregated Frames = 1), there are limited gains in changing from a PLDR of 150 Mbps to 450 Mbps. However, we see throughput nearly doubled and tripled when we also aggregate as many frames as

possible (i.e., # Aggregated Frames = 32). The combination of physical and MAC layer features leads to realized throughput that is many times greater than was possible with 802.11g. The focus and challenge in this thesis is in accurately simulating these features.

2.3.4 T-SIMn

T-RATE is limited to simulating 802.11g networks, which has only 8 rates, one for each Modulation and Coding Scheme (MCS). It does not support any of the 802.11n features described above. 802.11n is challenging to simulate due to the sheer number of possible Physical Layer Data Rates (PLDRs) available. There are a total of 128 PLDRs using the 8 different MCSs, 4 Spatial Streams (SSs), 2 Guard Intervals (GIs) and two available Channel Bandwidths (CBs). However, Frame Aggregation (FA) and Block Acknowledgment (BA) pose the greatest challenge in simulating 802.11n, with throughput being determined by many factors, including the pattern of failures in an aggregate frame and the number of frames that can be aggregated in a single frame. We name our new system T-SIMn, because the focus is no longer on the simulation of Rate Adaptation Algorithms (RAAs), but instead on the simulation of 802.11n networks in general. We describe T-SIMn and detail the changes required for 802.11n, in Chapter 3.

Chapter 3

T-SIMn Design and Implementation

T-SIMn is a trace-driven framework for simulating 802.11n networks with frame aggregation, that builds upon an existing framework, T-RATE, for 802.11g. We continue from the introduction of 802.11n, from Section 2.3, with the notation that we will use to concisely describe 802.11n rate configurations, in Section 3.1; and notation for describing frame aggregation length limits, during trace collection and simulation, in Section 3.2. We then provide an overview of the T-SIMn system, with specific emphasis on the differences between T-SIMn and T-RATE, in Section 3.3. We then describe the test bed, including the hardware and software, that we have available for evaluating T-SIMn. We close this chapter by describing the experimental methodologies that will be used in the remainder of this thesis.

3.1 Rate Configurations

In 802.11g, a transmission rate can be uniquely identified by the index of the Modulation and Coding Scheme (MCS). This index alone is no longer sufficient to uniquely identify an 802.11n rate, as a transmission rate is now also dependent on the the number of Spatial Streams (SSs), the Guard Interval (GI) and the Channel Bandwidth (CB). We refer to this set of parameters as a rate configuration. In the interest of brevity, we introduce the following notation for describing a rate configuration (sometimes simply referred to as a rate):

Format: (# of Spatial Streams)-(MCS Index)-(Guard Interval)-(Channel Bandwidth)=(Physical Layer Data Rate)

# of Spatial Streams: `1S`, `2S` or `3S`, for one, two and three SSs, respectively.

MCS Index: `I0`, `I1`, `I2` ..., `I7`, for MCS indices 0, 1, 2 ..., 7.

Guard Interval: `LG` or `SG`, for Long Guard Interval (LGI) or Short Guard Interval (SGI), respectively.

Channel Bandwidth: `20M` or `40M`, for 20MHz or 40MHz CB, respectively.

Physical-Layer Data Rate: `6.5`, `13`, ..., `405`, `450`, for 6.5 Mbps, 13 Mbps ..., 405 Mbps, 450 Mbps.

For example, `1S-I4-LG-20M=39` means 1 SS, MCS 4, LGI, 20 MHz CB and Physical Layer Data Rate (PLDR) of 39 Mbps; and `3S-I7-SG-40M=450` means 3 SSs, MCS 7, SGI, 40 MHz CB and a PLDR of 450 Mbps. This notation is used in the remainder of the document.

3.2 Frame Aggregation Length Notation

In order to more concisely describe limits on the length (i.e., the number of subframes) of an aggregated frame, we introduce the following notation:

$\texttt{FA}_{\texttt{(SIM | COL)}}$=*maximum number of aggregated subframes*

For example, $\texttt{FA}_{\texttt{SIM}}\texttt{=16}$ means that the simulator is allowed to aggregate a maximum of 16 subframes. Note that $\texttt{FA}_{\texttt{SIM}}\texttt{=16}$ does not mean that all aggregated frames in a simulation have exactly 16 subframes. The number of subframes in a specific aggregated frame may be further limited by the Block-Ack window and the rate configuration, discussed in detail in Section 4.2.2. Similarly, $\texttt{FA}_{\texttt{COL}}\texttt{=32}$ means that the driver was limited to aggregating a maximum of 32 subframes during trace collection. Again, $\texttt{FA}_{\texttt{COL}}\texttt{=32}$ does not mean that all aggregated frames in the collected trace have exactly 32 subframes, as aggregation may be limited by the factors discussed in Section 4.2.2. Furthermore, we use the notation $\texttt{FA}_{\texttt{SIM}}\texttt{=MAX}$ and $\texttt{FA}_{\texttt{COL}}\texttt{=MAX}$ to mean that we impose no further restrictions, beyond those in the 802.11 standard, on the number of subframes that may be aggregated during simulation and collection, respectively. In other words, $\texttt{FA}_{\texttt{SIM}}\texttt{=MAX}$ and $\texttt{FA}_{\texttt{COL}}\texttt{=MAX}$ signifies that we always aggregate as many subframes as possible. Lastly, we use $\texttt{FA}_{\texttt{SIM}}\texttt{=FA}_{\texttt{COL}}$ to mean that the limits on the length of an aggregated frame are the same during simulation and trace collection.

3.3 System Overview

Unlike T-RATE, which uses three phases, T-SIMn uses two phases to simulate 802.11n traces. The first phase is trace collection, where a log containing the data necessary for replaying an 802.11 experiment is collected. The second phase is simulation, where the trace is used to determine frame fates, transmission delays and throughput. We begin by discussing the methodology used for trace collection in Section 3.3.1. In Section 3.3.2, we describe the reasons for eliminating the trace preparation phase. Lastly in Section 3.3.3, we explain how the simulator uses the collected traces to simulate an 802.11n network, including computing frame fates and transmission delays, as well as performing Frame Aggregation (FA).

3.3.1 Trace Collection

As is the case for T-RATE, the purpose of trace collection in T-SIMn is to capture channel access and channel error rate information (i.e., a trace), for each 802.11n rate configuration. However, the implementation of the trace collection phase in T-SIMn differs considerably from T-RATE. Recall from Chapter 2 that T-RATE uses tcpdump to record outbound frame transmissions and their fates, as well as inbound transmissions to capture third-party WiFi traffic. T-RATE also uses a modified Ath9k driver that logs Cycle Counter Information (CCI) to capture non-WiFi traffic. Using two logs, one captured at the application layer (tcpdump) and one at the physical/MAC layer (Ath9k driver), is challenging because it requires corresponding frames to be matched between traces. This becomes even more difficult in 802.11n due to the addition of FA. Since tcpdump runs at the application layer, aggregated frames are represented as a series of individual frames, since FA is abstracted away from the application layer by the MAC layer. In T-SIMn, trace collection has been modified to obtain traces from a single source, a log from a modified Ath9k driver. This log captures both frame fate and CCI, which replaces the combination of a tcpdump log and a driver log in T-RATE. Using a new technique for differentiating delay due to WiFi interference from delay due to non-WiFi interference, CCI is now used to capture both delays. This technique is covered in detail in Chapter 5. For T-SIMn, we collect a single log entry for each aggregated frame, rather than for each individual subframe, as is done in T-RATE. This provides two key advantages: subframes do not need to be grouped into aggregate frames during post-processing, which is especially challenging when subsequent aggregate frames have the same rate configuration; it also significantly reduces the total number of log entries that are produced and thus reduces the size of the logs. In the initial implementation where individual frames were logged, at higher transmission rates

the sheer number of frames transmitted per second led to missing entries when the logging system, rsyslog [25], could not keep up with the volume. In the current implementation, with a single log entry produced per aggregated frame, we do not observe this problem, even at the highest transmission rate.

We modify the Ath9k driver to print a log entry after each aggregated frame transmission has completed, which also includes the reception of a Block Acknowledgment (BA) (or a timeout if no BA is received). To produce log entries, we use the Linux kernel function *printk* and prepend each line with the tag "`[AGGR]`". We use the system logging daemon rsyslog [25] to filter log entries from other kernel messages, using the "`[AGGR]`" tag, and to periodically write them to disk. An example log entry is shown in Figure 3.1. Note that we log the bitmap of subframe fates within the aggregated frame, as the pattern of failures can have a dramatic impact on throughput. This issue is examined in detail in Section 4.2.

```
[15550578.728446] [AGGR] 1 6 1 1 0 12 32 1 30 265863 2719
          270536 314515 2947 00000000fffe001f
```

Figure 3.1: Example log line for a frame in T-SIMn. Read from left to right, the frame was sent at kernel time `15550578.728446` (in seconds since system boot time), has the tag "`[AGGR]`" to signify an aggregated frame, was an 802.11n rate (`1`), used MCS `6`, SGI (`1`), 40 MHz (`1`), did not use Request-to-Send (RTS) (`0`), `12` subframes failed, `32` frames were included in the aggregated frame, a BA was received (`1`), the Received Signal Strength Indicator (RSSI) of the BA was `30` dBm, it took `265,863` clock cycles to transmit the data, `2,719` cycles to receive the BA, the device was busy for `270,536` cycles (transmitting, receiving or waiting for channel access) and the device took `314,515` cycles total (busy and idle) to transmit the frame, the sequence number was `2,947` and the subframe fate bit pattern (represented in hexadecimal) was `fffe001f` (with 1's meaning the subframe at that index was successfully received and 0's meaning the subframe failed).

To prepare for trace collection, we begin by creating an 802.11n Access Point (AP), using hostapd [24], on a desktop computer running our modified Ath9k driver. Trace collection must always be done on the *sender*. Although the AP could be used as the *sender* or the *receiver*, we use the computer designated as the AP as the *sender* in all experiments. The major advantage of this approach is that there are fewer requirements imposed on the *receiver*, which does not need to be capable of creating an AP. In addition, the *receiver*

does not need to use a modified Ath9k driver and, as a result, can be any 802.11n-capable device that runs Iperf3 [32]. For example, this allows us to perform experiments using an iPhone as a *receiver*. We then establish a network between the AP (*sender*) and a client, which acts as a receiver. To collect a trace, the *sender* saturates the link to the *receiver* by sending as many User Datagram Protocol (UDP) frames, of size 1,470 bytes, as possible using Iperf3 [32]. Unless otherwise noted, we aggregate the maximum number of subframes (i.e., $\texttt{FA}_{\texttt{COL}}\texttt{=MAX}$); see Section 3.2 for more details. During collection, the *sender* cycles through a set of 802.11n rates (shown in Appendix A), in a round-robin fashion. In constant rate trace collection, there is only one rate configuration in the set of round-robin rates. After each transmitted frame, the *sender* switches to the next rate in the set; wrapping back around after all rates have been sampled. This round-robin cycling of rates continues for the entire duration of the trace collection. After the desired amount of time has elapsed, the logs produced by the modified Ath9k driver are stored to disk. This log constitutes the trace that is passed on to the next phase.

Note that in T-RATE, all eight 802.11g rates were sampled but we now have 128 possible rate configurations in 802.11n (96 for our 3 antenna devices). Sampling the 96 rate configurations supported by our WiFi devices takes over 300 ms, which results in only a few samples within the channel coherence time. Trace collection for all 802.11n rates remains an open problem, which will be investigated in future work. However, we do show that the current implementation of T-SIMn is accurate when limiting trace collection to rate configurations with 1 antenna (i.e, 32 rate configurations). This evaluation of T-SIMn, including trace collection with all single antenna rate configurations (32 rates), is presented in Chapter 7.

3.3.2 Trace Preparation

Recall from Section 2.2.2 that the trace preparation phase of T-RATE pre-computes frame fates for all 802.11g transmission rates and for all discretized times within the bounds of the trace. In T-SIMn, we make the decision to eliminate the trace preparation phase and the collected trace is now the direct input to the simulator. With 96 possible rates, each with 32 Subframe Index Error Rates (SFIERs) (discussed in Section 6.1), it becomes computationally expensive to pre-compute all frame fates and the resulting traces would be prohibitively large. It would also have resulted in much wasted work. For example, we transmit at a single rate at any given time, meaning that the pre-computed frame fates for the remaining 95 rates would be unused. Moreover, we find eliminating this phase better facilitates iterative development of SIMn. This new approach can be thought of as Just In Time (JIT) virtual trace preparation.

Despite eliminating the trace preparation phase, many techniques have been carried forward from T-RATE. To compute the frame fate at time t, rate r and subframe index i, T-SIMn considers all samples for the rate r and subframe index i in the averaging window ($t - \frac{window\_size}{2}$ ms to $t + \frac{window\_size}{2}$ ms), from the collected trace. The averaging window should be less than the channel coherence time for the environment so that channel conditions are relatively constant with respect to the frames being used to determine the fate of packets at time t. Channel coherence time is dependent on many factors, such as movement speed (i.e., longer in stationary environments and shorter in high speed environments) and channel frequency (i.e., longer for 2.4 GHz channels and shorter for 5 GHz channels). In indoor environments at walking speeds, channel coherence time is reported to be approximately 200 ms for the 2.4 GHz band [31] and 100 ms for the 5 GHz band [5]. Because all of the experiments conducted to evaluate T-SIMn use the 5 GHz band, we use an averaging window of 200 ms (i.e., $t - 100$ ms to $t + 100$ ms) which corresponds to a coherence time of 100 ms. Our evaluation in Section 7 shows that an averaging window of 200 ms produces accurate results in our mobile environment at walking speeds, when performing round-robin trace collection with 1 antenna. However, this parameter is easily set in SIMn so that it can be tailored to other environments.

To compute a Subframe Index Error Rate (SFIER), the number of these samples that have failed (i.e., subframes that are not acked) is divided by the total number of these samples. A uniform random number is then drawn, in the range [0,1.0]. A subframe at time t, rate r and index i is considered successful if the random number is larger than the error rate. The two major differences from T-RATE are that the functionality of the trace preparation phase is now moved to the simulation phase (called trace processing in T-RATE); and that SFIERs are now considered.

3.3.3 Simulation

The trace processing phase uses the collected trace to simulate other experiments. Similarly to T-RATE, SIMn performs a time based simulation using an 802.11n trace, where the *sender* saturates the link to the *receiver* by sending as many aggregated frames as possible. SIMn currently simulates constant rate or round-robin experiments, and aggregates subframes using the default Frame Aggregation Algorithm (FAA) of the Ath9k device driver. However, SIMn is designed to allow porting, developing, evaluating and comparing new Rate Adaptation Algorithms (RAAs) and FAAs. We also expect that SIMn will be a valuable tool for studying 802.11n configuration options such as CB selection (20 or 40 MHz) and GI selection. We discuss these avenues for future research in Section 8.3.

A simulation in SIMn proceeds using an event loop, starting a time $t = 0$, that performs the following steps:

1. We check the collected trace for unprocessed WiFi delays that occurred before time t. If any such WiFi delays exist, t is incremented by the duration of the delay. This is repeated until all WiFi delays that occurred before t have been processed. Computation and differentiation of WiFi delays is discussed in Section 5.1.

2. We compute non-WiFi delay at time t and increment t by the duration of the delay.

3. If there are fewer than two aggregate frames (one in transmission and one queued), we form an aggregate frame (ensuring that no aggregated frame length limits are violated) and add it to the queue. Aggregated frame queuing, and length limits, are analyzed in Section 4.2.2.

4. Transmission time for the aggregated frame is computed and t is advanced.

5. The fate of each subframe i in the aggregate frame is determined using the SFIER at time t, using rate r and index i. Successful subframes increment a packet counter, which is used to report throughput during the desired simulation time interval. Failed subframes are rescheduled for transmission if they have not reached their retry limit. Rescheduled frames are given priority when forming aggregate frames. SFIERs are discussed in greater detail in Section 6.1.

As in T-RATE, this process repeats until the simulation ends; this is generally after the elapsed simulation time is equal to the duration of the collected trace. Figure 3.2 provides an overview of this process. The two primary points for modification to support future research are obtaining a rate configuration from an RAA (discussed in Section 8.3.2) and the Frame Aggregation Algorithm (discussed in Section 8.3.3).

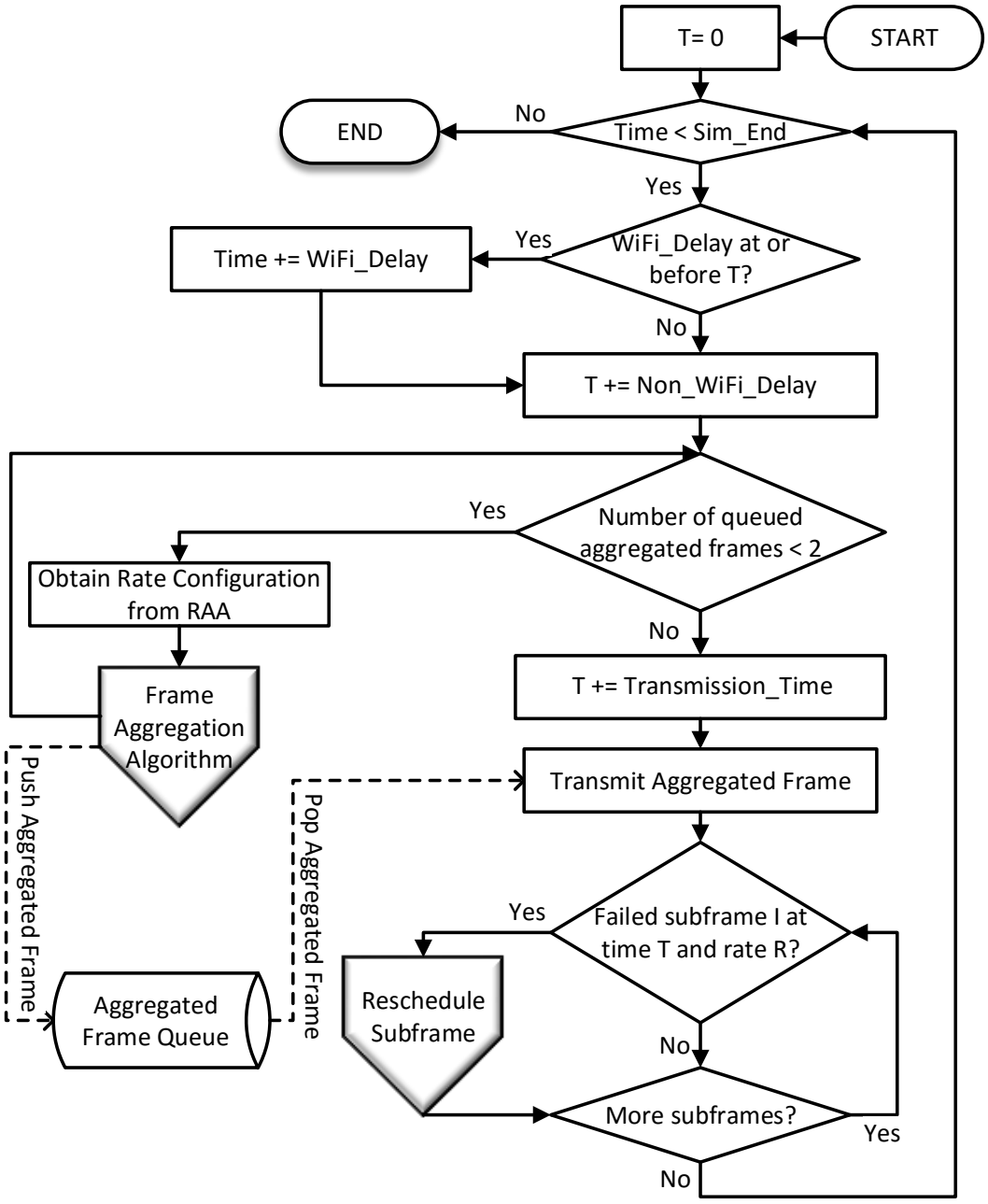

Figure 3.2: T-SIMn Simulation Flowchart. Solid lines represent the execution flow of the simulator, with the direction being indicated by a closed (i.e., filled) arrow. Dashed lines indicate the flow of data, with the recipient of the data being indicated with an open arrow. "Form Aggregated Frame" and "Reschedule Subframe" are entry points to the process of forming aggregated frames, shown in Figure 3.3.

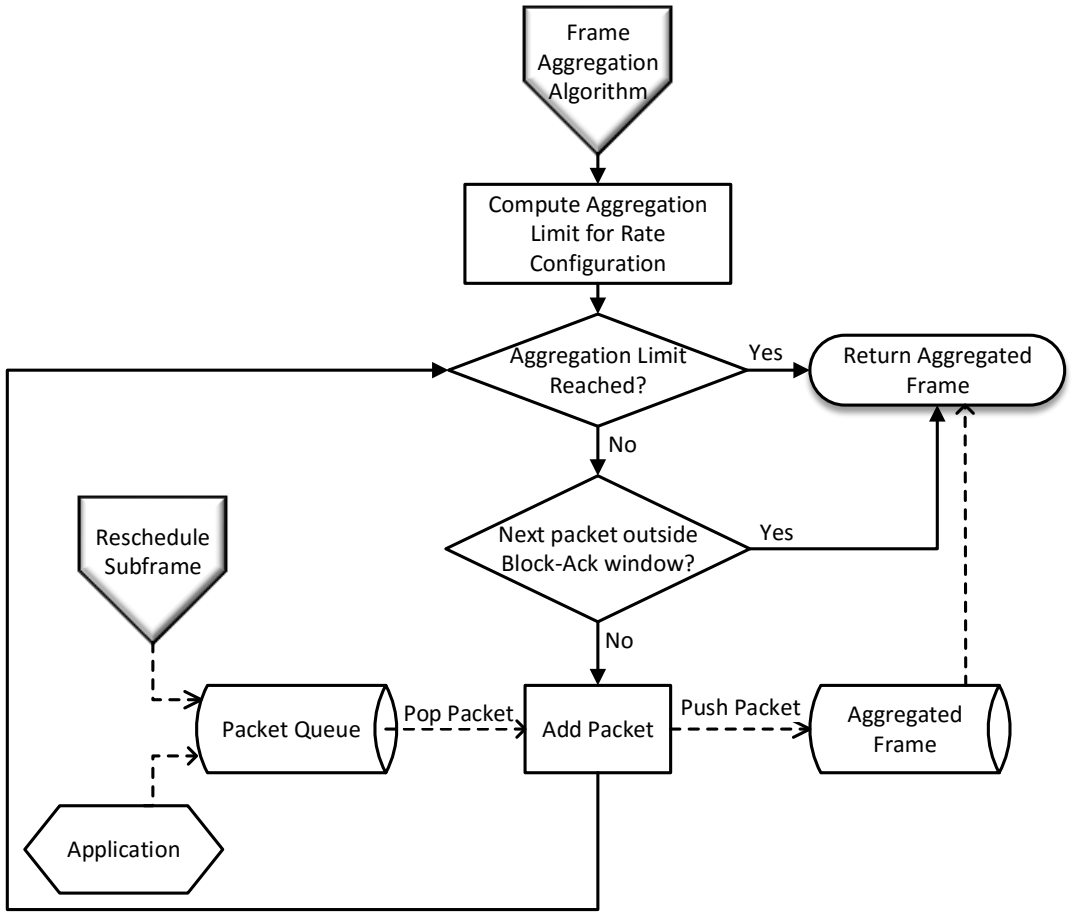

Figure 3.3: T-SIMn Aggregated Frame Formation Flowchart. Solid lines represent the execution flow of the process, with the direction being indicated by a closed (i.e., filled) arrow. Dashed lines indicate the flow of data, with the recipient of the data being indicated with an open arrow. The entry point of this process is "Form Aggregated Frame", which is called in Figure 3.2. The packet queue is populated with frames from a simulated application or from failed subframes, from Figure 3.3, that have been rescheduled for transmission. The process ends by returning an aggregated frame to the T-SIMn simulator.

3.4 Test Bed

For conducting experiments, we have created a test bed in our lab on the university campus. This environment consists of cubicles in a large room, as well as some separate offices. Our test bed consists of three desktop computers housing identical TP-Link TL-WDN4800 PCIe cards, an Apple MacBook Pro (Retina, 15-inch, Mid 2012) and an Apple iPhone 6. The TP-Link cards use an Atheros AR9380 chipset, while the MacBook uses a Broadcom BCM4331. All five devices are dual-band (2.4 and 5 GHz). All contain three antennas which support three spatial streams in a 3x3:3 MIMO configuration, except the iPhone 6 which contains only one antenna. We use one of the desktop computers as an AP, which acts as the *sender* in all experiments. This computer will be referred to as the "the AP" in the remainder of the document. We use the remaining two desktop computers as *receiver*s. One is located in close proximity (less than 1 meter) to the AP, with line of sight; while the other is further away (roughly 10 meters) in a separate office, without line of sight due to cubicles and a wall. We refer to these *receiver*s as the "Near Client" and the "Far Client", respectively. The MacBook and iPhone are used only for mobile experiments and we refer to them by name. In order to minimize the amount of uncontrolled WiFi interference, we run all experiments in the 5 GHz band on channels that are unused by other APs in the vicinity, unless otherwise noted. These channels are also subject to interference from fewer non-WiFi devices, as most devices operate in the 2.4 GHz band. We monitor for interference using an AirMagnet Spectrum XT [4] spectrum analyzer. For generating controlled non-WiFi interference, we have an RF Explorer Handheld Signal Generator [26] (RFE6GEN), which we control using a USB connection.

The three desktop computers run Ubuntu 12.04 with Linux kernel version 3.13.0. All three use a modified version of the Ath9k driver provided by the backports-3.14-1 package; a set of drivers backported from Linux kernel 3.14-1, in order to run on earlier versions of the kernel. The MacBook runs OS X 10.11.1 El Capitan and the iPhone runs iOS 9.2, both using the default drivers provided by Apple. We use hostapd [24] v0.7.3 to create an Access Point (AP) that the three client computers can connect to. iPerf3 [32] is used to saturate the link between the *sender* (AP) and a *receiver*. By using a single desktop computer as the AP and the *sender*, the *receiver*s do not need to be capable of running the modified Ath9k driver and can be any device capable of running iPerf3, such as the MacBook. This will allow other devices, such as cell phones and tablets, to be used in the future. To capture the logs that we produce in the Ath9k driver, we use rsyslog [25].

3.5 Experimental Methodologies

In this section we describe the methodologies that are used to evaluate the simulator, SIMn, and the full T-SIMn framework. As the focus of this thesis is the simulator component of the T-SIMn framework, we evaluate SIMn using a technique that minimizes the reliance on the trace collection methodology. We conduct an experiment using a constant rate configuration, which produces a constant rate configuration trace. We then use SIMn to simulate a constant rate configuration experiment (for the same rate configuration) using the trace. Since the simulated experiment uses the same rate configuration as the conducted experiment, simulated throughput should match throughput obtained during the conducted experiment, if the simulator is accurate. There are two major advantages to this methodology:

1. It does not require experiments to be repeatable since the experiment produces a trace that is used by SIMn to simulate an experiment with exactly the same conditions and environment as the conducted experiment (i.e., the conducted experiment *is* trace collection). For comparison, if experimentation and trace collection were two separate steps, it would be challenging to ensure that the exact same conditions and environment are maintained for both the experiment and trace collection.

2. Constant rate traces provide many samples in each averaging window, which means that the accuracy of SIMn is not limited by the collected trace.

Together, these properties allow us to expect, and check for, close matches between experimental and simulated throughput when evaluating SIMn. This facilitated iterative improvements to the accuracy of SIMn, as we did not need to determine if a mismatch was due to an inaccuracy in the simulator or due to a lack of experimental repeatability or a lack of samples in the collected trace. Note that we show 95% confidence intervals in all plots in this thesis and we consider a match to be overlapping confidence intervals for experimental and simulated throughput. We use this approach to evaluate SIMn's accuracy in simulating the new physical and MAC layer features in 802.11 in Chapter 4, channel access delays in Chapter 5 and error rates in Chapter 6.

However, in order to evaluate the T-SIMn framework, we use an evaluation methodology similar to that used to evaluate T-RATE [2]. That is, we conduct an experiment (which produces a trace) using a round-robin ordering of rate configurations and then in SIMn we conduct a simulation using a round-robin ordering that differs from the experiment. This is done to verify the averaging window approach to determining a frame fate at time

t in SIMn, when the simulated rate configuration differs from that in collected trace at time t. In contrast to T-RATE, we cannot cycle through all of the available 802.11n rates when performing round-robin trace collection, due to the time required to perform a full round (roughly 300 ms). We instead limit trace collection to the set of 1-Spatial Stream (SS) rates, using Long Guard Interval (LGI), Short Guard Interval (SGI), 20 and 40 MHz Channel Bandwidth (CB), for a total of 32 rates. We choose this set because its use is prevalent in cell phones, tablets and other small devices, most of which contain only 1 antenna. We use this methodology in Chapter 7 to demonstrate that, despite not being able to collect traces for 2 or 3 SSs, the full framework is practical for 1 SS.

In many cases, we collect "error-free" traces. However, in practice, it is impractical to collect truly error-free traces due to the nature of wireless transmission being less reliable than wired. As a result, we define error-free to be an error rate of less than 0.1%.

Scenarios

We use the general term *scenario* to describe a set of events or the environment in which a trace is collected. For example, a mobile scenario refers to trace collection with a moving *receiver* and an error-free scenario means that trace collection is performed such that there is no path loss or error due to interference. Note that we define "error-free" to be an error rate of less than 0.1% in each reported time interval, as it is impractical to collect truly error-free traces due to the nature of wireless transmission being less reliable than wired.

Accuracy and Realism

Although we strive to collect traces that allow the trace preparation phase to accurately reflect the channel conditions experienced in reality, this is not required for comparisons of different algorithms using T-SIMn. Even if there are slight inaccuracies with respect to reality, all algorithms are exposed to the same inaccuracies and thus the comparison is still viable and, most importantly, fair. This also allows us to synthetically generate traces that would be difficult to collect experimentally. We use this technique to study Subframe Index Error Rates (SFIERs) in Section 6.1.

The described trace collection details and experimental methodologies will be used in the remaining chapters; any deviations will be explicitly mentioned in the experimental setup. We now evaluate the simulation of physical and MAC layer features of 802.11n in Chapter 4; channel access delays in Chapter 5; and channel error rates in Chapter 6.

Chapter 4

Simulating 802.11n Features

802.11n introduces both physical and MAC layer enhancements to increase throughput, compared to 802.11g. We consider each of the major features of 802.11n and, specifically, how they impact simulation. We discuss the physical layer enhancements, which focus on increasing the raw data rate of transmission, in Section 4.1. We then examine the MAC layer enhancements, in Section 4.2, which instead focus on improving the efficiency of the 802.11 protocol.

4.1 Physical Layer Features

The 802.11n protocol introduces many new physical layer features, namely Multiple Spatial Streams (SSs), Short Guard Intervals (SGIs), Channel Bonding and Dual-Band support. Despite being numerous, these features are comparatively simpler to simulate than the MAC layer features and did not require major architectural changes to the simulator. However, we discuss each of these features in detail because they required precise changes to the simulator. In this section we focus on timing, rather than interference and transmission errors, which are studied in Chapters 5 and 6, respectively. We now evaluate SIMn's ability to accurately replicate the timings, and consequently the throughput, of these 802.11n physical layer features.

Experiment Setup:

We create a network between the Access Point (AP) (*sender*) and a desktop client (*receiver*). We use the Near Client because it can reliably obtain error-free traces, due

to its close proximity and line of sight. Recall from Section 3.5 that we define error-free to be an error rate of less than 0.1%. We collect 100 second traces for the rate configurations `1S-I4-LG-20M=39`, `1S-I4-SG-40M=90`, `2S-I4-SG-40M=180` and `3S-I4-SG-40M=270`. We choose these rate configurations because they cover both long and short Guard Intervals (GIs); 20 and 40 MHz Channel Bandwidths (CBs); as well as one, two and three spatial streams. Modulation and Coding Scheme (MCS) Index 4 is chosen because it is the highest index with which we could reliably obtain error-free traces. We use high rates because discrepancies between experimental and simulated throughput are more likely to be seen at high rates than low rates. We simulate constant rate experiments for each of the chosen rate configurations, using the collected traces as input to the simulator. We expect simulated throughput to match experimental throughput obtained from the collected traces, for each of the respective rate configurations.

In Figure 4.1, we plot pairs of throughput measurements, simulated and experimental, for each of the collected rate configurations. We analyze the results in the following sections.

4.1.1 Multiple Spatial Streams (MIMO)

Multiple Spatial Streams allow multiple data streams to be transmitted simultaneously using different antennas, one for each stream. Using two or three SSs exactly doubles or triples, respectively, the Physical Layer Data Rate (PLDR) of a one stream configuration. At the physical-layer, both individual and aggregated frames have a header that is defined by the Physical Layer Convergence Protocol (PLCP) [12]. The PLCP header includes several training fields that are used for tuning the receiver. One type of training field is the High-Throughput Long Training Field (HT-LTF), which aids in decoding MIMO transmissions [12]. The number of SSs that are used for transmission affects the number of HT-LTFs that must be included in the PLCP header. For one and two SSs, 1 and 2 HT-LTFs are needed, respectively; for three or four SSs, 4 HT-LTFs are needed [12] (though our WiFi cards support a maximum of three SSs). These training fields increase the size of the PLCP header, which consequently increases the duration of time required for transmission. To accurately simulate multiple SSs, the simulator must consider both the PLDRs and the PLCP duration, in terms of the number of streams.

We show that SIMn can accurately handle one, two and three SS by examining the rate configurations `1S-I4-SG-40M=90` (one stream), `2S-I4-SG-40M=180` (two streams) and `3S-I4-SG-40M=270` (three streams), in Figure 4.1. These rate configurations differ only by the number of SSs and cover the three possible values. For each configuration, there is a tight match between simulated and experimental throughput. This suggests that SIMn

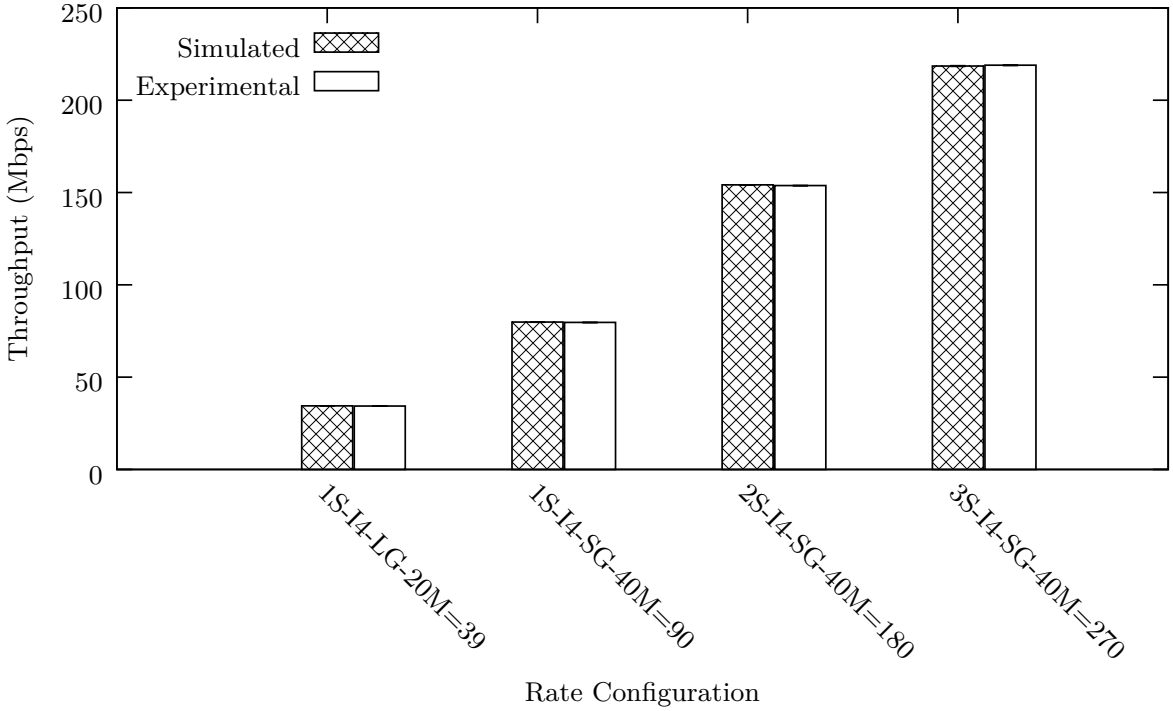

Figure 4.1: Simulated throughput using the new physical-layer features in 802.11n (SGI, 40 MHz and Multiple SSs) compared against throughput measurements obtained experimentally.

accurately simulates one, two and three spatial stream rate configurations in an error-free scenario. Figure 4.1 also demonstrates how throughput scales as the number of SSs is increased, with the throughput of `2S-I4-SG-40M=180` (154 Mbps) being roughly double that of `1S-I4-SG-40M=90` (80 Mbps). Similarly, throughput of `3S-I4-SG-40M=270` (219 Mbps) is nearly triple that of `1S-I4-SG-40M=90` (80 Mbps). Although the PLDRs exactly double or triple using two or three SSs, respectively, throughput does not quite double or triple due to overheads inherent in the 802.11 protocol.

4.1.2 Short Guard Interval (SGI)

SGI decreases the duration of time between the transmission of successive data symbols, from 800 ns to 400 ns. This decreases the time to transmit each symbol from 4 μs to 3.6 μs, which in turn decreases the time required to transmit a given frame. Using SGI

increases the Physical Layer Data Rate rate by up to 11%, compared to the corresponding Long Guard Interval (LGI) rate configuration.

Figure 4.1 also shows that SIMn accurately handles both GIs. This can be seen by examining the rate configurations `1S-I4-LG-20M=39` (LGI) and `1S-I4-SG-40M=90` (SGI). In both cases, the simulated throughput closely matches the experimental throughput, indicating that in error-free environments, SIMn accurately simulates both LGI and SGI rate configurations.

4.1.3 Channel Bonding

802.11n supports bonding two adjacent 20 MHz channels for a total channel width of 40 MHz. Channel bonding increases the Physical Layer Data Rate to slightly more than double that of the equivalent 20 MHz rate configuration. This increases the number of bits per symbol, which decreases transmission time for a given frame. Additionally, during simulation we must consider the configuration of the AP that was used for trace collection. An AP supporting 40 MHz channel widths may use a higher clock frequency for the Analog-to-Digital Converter (ADC), as is the case with our devices. This impacts the delay calculations, which rely on the Cycle Counter Information (CCI), and are discussed in greater detail in Chapter 5.

We verify that the simulator handles both 20 and 40 MHz channel widths by studying the rate configurations `1S-I4-LG-20M=39` (20 MHz) and `1S-I4-SG-40M=90` (40 MHz), in Figure 4.1. In both cases, the simulated throughput tightly matches the experimental throughput. This suggests that SIMn accurately handles rate configurations using both 20 and 40 MHz channel widths.

4.1.4 Dual Bands

The 802.11n protocol supports both the 2.4 and 5 GHz bands. These bands have different signal propagation and interference characteristics, but the choice of band does not affect the PLDR. However, the duration of Short Inter-Frame Space (SIFS), and consequently Distributed Inter-Frame Space (DIFS), depends on the band being used. The duration of DIFS is $SIFS + 2 \times Slot\ Time\ \mu s$. The durations for SIFS, Slot Time and DIFS are shown in Table 4.1, for both 2.4 and 5 GHz. Since these timings differ, we determine whether the 2.4 or 5 GHz band was used during trace collection, and use the respective SIFS and DIFS values during simulation. However, we use the 5 GHz band for all experiments due to heavy utilization of all 2.4 GHz channels on the university campus.

Table 4.1: Timing Constants for Dual-Band 802.11n

Band (GHz)	SIFS (μs)	Slot Time (μs)	DIFS (μs)
2.4	10	9	28
5	16	9	34

4.1.5 Summary

We find that SIMn is able to match throughput obtained from collected traces with the rate configurations `1S-I4-LG-20M=39`, `1S-I4-SG-40M=90`, `2S-I4-SG-40M=180` and `3S-I4-SG-40M=270`, which cover both long and short GIs; 20 and 40 MHz CBs; as well as one, two and three spatial streams. This suggests that SIMn accurately simulates the new physical layer features in 802.11n.

4.2 MAC Layer Features

802.11n introduces two major features at the MAC layer, Block Acknowledgment (BA) and Frame Aggregation (FA), that aim to improve the efficiency of the protocol. In the following two sections we describe Block Acknowledgment and Frame Aggregation, describing how they work together to increase throughput in 802.11n networks.

4.2.1 Block Acknowledgments

In an 802.11g (or earlier) network, each successfully received frame is immediately acknowledged with an Ack frame. Like other management frames, Acks are transmitted using a rate from the basic rate set, which is the set of rates that all members of the network must support. It is most common for 802.11n networks to support a mixed mode [12], which retains backwards compatibility with 802.11a, b and g devices. This means that the basic rate set is typically limited to at most 11 Mbps (i.e., the maximum 802.11b rate) for 2.4 GHz networks and 54 Mbps (i.e., the maximum 802.11a rate) for 5 GHz networks. These Acks result in significant inefficiencies due to the time required for their transmission, relative to the time required for the data frame transmission, especially in 802.11n networks with Physical Layer Data Rates (PLDRs) of up to 450 Mbps for data frames. To improve the efficiency of the protocol, 802.11n adds Block Acknowledgment (BA) as a mandatory feature [12], which allows multiple frames to be acknowledged at once. 802.11n

networks use a type of BA called a Compressed Block-Ack, which contains a starting sequence number and a 64-bit bitmap that represents the fate of up to 64 frames in the range, $starting\_sequence\_number$ to $starting\_sequence\_number + 64 - 1$. This range is called the Block-Ack Window (BAW) and is established with a Block-Ack Request (BAR). The BAW is the range of sequence numbers that the *receiver* is expecting and frames falling outside this window will be ignored. After a *receiver* responds with a BA, both the *sender* and *receiver* advance their BAW to the sequence number of the last frame for which all preceding frames have been acknowledged. For example, if the starting sequence number of the BAW is 0 and the *sender* transmits frames 0, 1, 2 and 3, but only 0, 1 and 3 are received, the starting sequence number of the BAW will be advanced to 2. BAs are especially useful in combination with Frame Aggregation (FA), which we describe in the following section.

4.2.2 Frame Aggregation

Frame Aggregation (FA) is a new feature in 802.11n that allows multiple frames to be combined to form a larger frame. By sending these frames together, as an aggregated frame, the *sender* only needs to perform channel sensing, backoff, Distributed Inter-Frame Space (DIFS), Short Inter-Frame Space (SIFS) and wait for an Ack once for the entire set of frames, rather than for each individual subframe. This results in a greater proportion of time being spent transmitting data, increasing the airtime efficiency. Recall from Figure 1.1 that throughput without frame aggregation (i.e., sending individual frames), does not increase significantly as the Physical Layer Data Rate (PLDR) is increased from 150 Mbps to 450 Mbps. As the PLDR increases, the duration of frame transmission gets smaller and smaller. However, the wait times listed above do not decrease alongside the increases in PLDR, and eventually the transmission time becomes insignificant relative to the wait times. As shown in Figure 1.1, throughput gains (in an error-free scenario) are larger when we aggregate more frames, though this too reaches a point of diminishing returns. Note that in Figure 1.1, the number of frames being aggregated is increasing exponentially but we see roughly linear increases in throughput. We refer to the number of subframes in an aggregated frame as the frame length. Generally, longer aggregated frames result in higher throughput, although this may not always be the case. There are many factors that limit the length of an aggregated frame:

- There is an airtime limit of 4 ms in the 802.11n standard that prevents a single device from monopolizing the channel by aggregating frames. This airtime limit means that when using slower rate configurations (i.e., lower PLDRs), aggregated frame length may be limited by the number of frames that can be transmitted in 4 ms.

- The Block-Ack Window (BAW), described in the previous section, also plays a major role in the length of aggregated frames, as the sequence numbers of subframes can be offset by at most 64 from the starting sequence number in the BAW.
- There is a 64 KB limit on the size of a User Datagram Protocol (UDP) packet. For example, if using 1,500 byte frames, a maximum of 43 frames may be aggregated.
- Implementation specific requirements may also limit the length of aggregated frames. For instance, the Ath9k driver used in this thesis imposes a limit of 32 subframes in an aggregate frame, though this is not dictated by the 802.11n standard. This is done so that another aggregated frame can be constructed and queued while the first is in transmission. A limit of 32 subframes in each aggregate frame means that two aggregates frames can be created within the 64-bit BAW, one in transmission and one queued.
- Management frames must be transmitted as individual frames. When a time sensitive management frame, such as a beacon frame, is queued, the Ath9k driver will stop aggregating subsequent frames. This allows the management frame to begin transmission sooner than if a long aggregated frame were ahead of it in the transmission queue.

During trace collection, we always aggregate as many subframes as possible (i.e., $\texttt{FA}_{\texttt{COL}}\texttt{=MAX}$, which is $\texttt{FA}_{\texttt{COL}}\texttt{=32}$ in Ath9k). Since we do not collect $\texttt{FA}_{\texttt{COL}}\texttt{=1}$, $\texttt{FA}_{\texttt{COL}}\texttt{=2}$, ..., $\texttt{FA}_{\texttt{COL}}\texttt{=MAX}$, we often need to simulate cases where $\texttt{FA}_{\texttt{SIM}} < \texttt{FA}_{\texttt{COL}}\texttt{=MAX}$, due to the many reasons listed above. We now evaluate SIMn's accuracy when simulating the throughput of frames aggregated with fewer subframes during simulation than were obtained during trace collection.

Experiment Setup:

We create a network between the Access Point (AP) (*sender*) and a desktop computer client in close proximity (*receiver*). We use the Near Client because it can reliably obtain error-free traces, for rate configurations with higher PLDRs than the Far Client (we study simulation with error in Chapter 6). For all experiments, the *sender* is set to a constant rate configuration of `2S-I4-SG-40M=180`. We collect a 100 second trace with $\texttt{FA}_{\texttt{COL}}\texttt{=MAX}$, as this is the aggregation limit that we will typically use for trace collection. We then conduct 100 second experiments with FA limited to 32, 16, 2 and 1 aggregated frames, which we use as the ground truth. We then simulate constant rate scenarios for the rate configuration `2S-I4-SG-40M=180` with $\texttt{FA}_{\texttt{SIM}}\texttt{=MAX}$, $\texttt{FA}_{\texttt{SIM}}\texttt{=16}$, $\texttt{FA}_{\texttt{SIM}}\texttt{=2}$ and $\texttt{FA}_{\texttt{SIM}}\texttt{=1}$, using the $\texttt{FA}_{\texttt{COL}}\texttt{=MAX}$ trace as input to the simulator. We expect simulated throughput for

$\texttt{FA}_{\texttt{SIM}}$=MAX, $\texttt{FA}_{\texttt{SIM}}$=16, $\texttt{FA}_{\texttt{SIM}}$=2 and $\texttt{FA}_{\texttt{SIM}}$=1 to match the throughput obtained directly from the experiments run with FA limited to 32, 16, 2 and 1 aggregated frames, respectively.

In Figure 4.2, we plot pairs of simulated and experimental throughput measurements, for each of the frame aggregation configurations. For all pairs of simulated and experimental throughput, we see a close match which suggests that SIMn can accurately simulate shorter aggregated frames from traces with $\texttt{FA}_{\texttt{COL}}$=MAX, in an error-free environment. Because SIMn can accurately simulate $\texttt{FA}_{\texttt{SIM}}$=1, $\texttt{FA}_{\texttt{SIM}}$=2, ..., $\texttt{FA}_{\texttt{SIM}}$=MAX from a trace collected with $\texttt{FA}_{\texttt{COL}}$=MAX, we do not need to do round-robin trace collection for $\texttt{FA}_{\texttt{COL}}$=1, $\texttt{FA}_{\texttt{COL}}$=2, ..., $\texttt{FA}_{\texttt{COL}}$=MAX. Instead, we always collect traces with $\texttt{FA}_{\texttt{COL}}$=MAX.

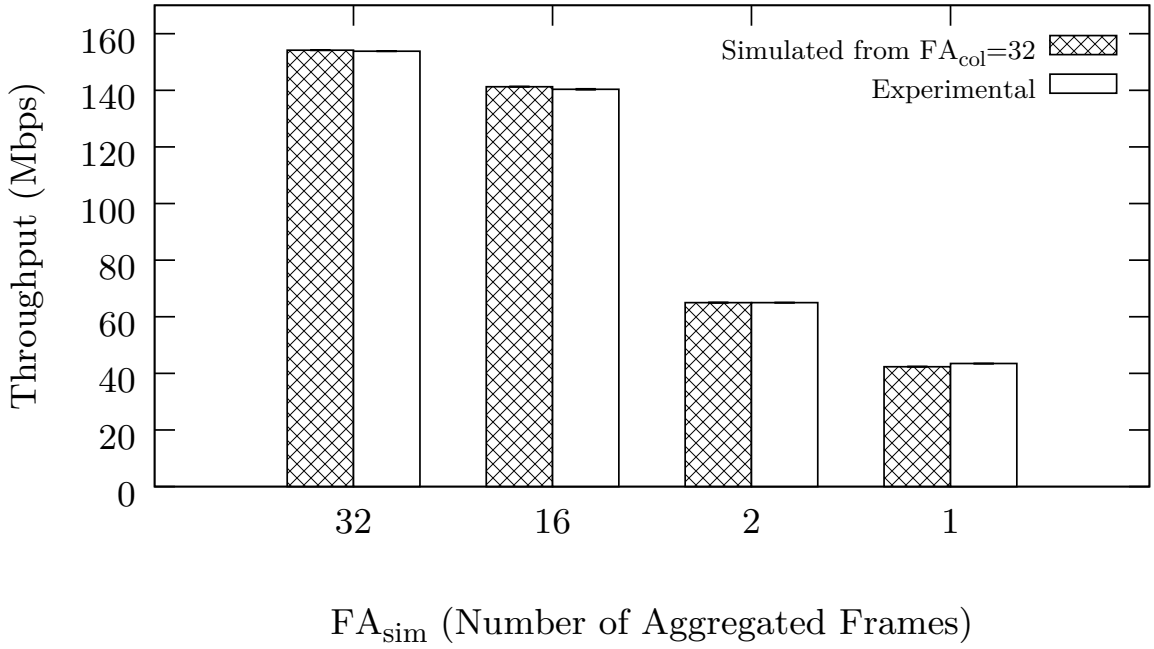

Figure 4.2: Experimental throughput for the rate configuration 2S-I4-SG-40M=180 with FA limited to 32, 16, 2 and 1 aggregated frames, compared against simulated throughput for the rate configuration 2S-I4-SG-40M=180 with $\texttt{FA}_{\texttt{SIM}}$=32,16,2,1, using an $\texttt{FA}_{\texttt{COL}}$=32 trace as input to the simulator (i.e., simulating throughput when aggregating fewer frames during simulation than collection).

We do not repeat this evaluation in environments that are experiencing error due to interference or path loss because of a lack of experimental repeatability. However, simu-

lation of shorter aggregated frames from traces with $\texttt{FA}_{\texttt{COL}}$=MAX, in environments that are experiencing error, is indirectly tested in Section 6.2. The tight match between simulated and experimental throughput in Figure 6.2 suggests that shorter frames are accurately simulated in an environment with high error rates due to path loss. Developing a new experimental methodology for a direct evaluation of the simulation of shorter frames in these environments is proposed as future work, in Section 8.3.3.

4.2.3 Summary

Frame Aggregation (FA) and Block Acknowledgment (BA) are MAC layer features of 802.11n networks that are necessary to take advantage of the new physical layer features. These MAC layer and physical layer features can be combined to obtain throughput that is well beyond what can be achieved in 802.11g networks. We demonstrate that SIMn in an error-free environment can accurately simulate frame aggregation when aggregating as many frames as possible, $\texttt{FA}_{\texttt{SIM}}$=MAX, as well fewer frames, using a single $\texttt{FA}_{\texttt{COL}}$=MAX trace.

Chapter 5

Simulating Channel Access

Since WiFi networks use a shared medium for communication, channel access is a crucial factor in determining throughput for an experiment. 802.11n operates in both the 2.4 and 5 GHz bands, which are shared by other WiFi devices, such as computers, cell phones and tablets; as well as Bluetooth devices, wireless keyboards/mice, cordless phones, microwave ovens and many others. 802.11 is a *listen-before-talk* protocol [12], meaning that an 802.11 device will check that the channel is idle before transmitting, in order to avoid collisions. This process is called Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). In order to proceed with transmission, the channel must remain idle (Carrier Sensing) for the duration of a Distributed Inter-Frame Space (DIFS) (durations shown in Table 4.1) and a random backoff time. The random backoff time prevents multiple WiFi devices from transmitting immediately after the duration of a DIFS, resulting in a collision. To backoff, the device draws a uniform random number n between 0 and 15 and waits for $n \times Slot\ Time$ (from Table 4.1) μs. If the channel does not remain idle then the device must backoff again (Collision Avoidance), for an exponentially increasing duration of time, before re-checking the channel for activity. It is also possible for two or more devices to draw the same random number and begin transmitting at the same time, resulting in a collision (i.e., neither frame is decodable). The channel may also be occupied by non-WiFi devices, which are not required to follow this protocol and may broadcast on the channel at any time. If a non-WiFi device begins broadcasting before a WiFi transmission has begun, this will introduce a delay for the WiFi transmission because the channel will be sensed as not being idle. If a non-WiFi device begins broadcasting while a WiFi device is already transmitting, this will increase the likelihood that the transmission fails, due to overlapping signals. In this section we focus on the scenarios where WiFi and non-WiFi

interference results in delay, rather than error, though we cannot study the two in complete isolation.

In T-RATE [2], non-WiFi delay is computed using the Cycle Counter Information (CCI) available in the Ath9k driver; while WiFi delay is computed using a tcpdump trace that captures third-party WiFi packets. In T-SIMn, we now compute both WiFi and non-WiFi delay using the CCI, with a heuristic proposed by Ali Abedi for distinguishing the two types of delays [1]. The four available counters are *tx_ cycles*, which counts the number of cycles that the chip spends performing transmissions (outbound); *rx_ cycles*, which counts clock cycles spent receiving (inbound); *busy_ cycles*, which measures the number of cycles that the channel was busy performing transmission, receiving WiFi frames or due to non-WiFi noise; and *total_ cycles*, which counts the total number of cycles for transmission, including those spent busy (i.e., waiting). We modify the driver to report cycle counts after each aggregated frame, rather than for each subframe, because carrier sensing and backoff (delay) are not performed between subframes. We compute delay with the following general formula:

$$delay = actual\ duration - expected\ duration \tag{5.1}$$

The *actual duration* is the measured amount of time that the device spent transmitting the aggregated frame, according to the cycle counters. While we initially thought that the cycle counters were in reference to the WiFi card's chipset clock speed, they are instead measured in terms of the sampling rate of the Analog-to-Digital Converter (ADC) [23]. The clock speed is dependent on the maximum supported channel bandwidth of the Access Point (AP), which may be either 20 or 40 MHz in 802.11n. Our wireless cards use a clock speed of 44 MHz when the AP is limited to a 20 MHz Channel Bandwidth (CB) and 88 MHz for a CB of 40 MHz. This is consistent with the Nyquist rate [33] of $2 \times signal\ bandwidth$, which is a lower-bound on the sampling rate needed to avoid aliasing. The formula used for computing *actual duration* from the cycle counters is:

$$actual\ duration = \frac{total\_cycles}{clock\_speed} \tag{5.2}$$

For simplicity, we always enable support for 40 MHz channel widths (even though they are not necessarily used in all experiments) in the AP and thus always have a clock speed of 88 MHz.

The *expected duration* is the amount of time that we would expect the device to spend transmitting the aggregated frame in the absence of WiFi or non-WiFi interference. We can accurately compute the transmission time of both the data frame and the resulting ack frame, based on the number of bits per symbol for the given Modulation and Coding Scheme

(MCS) and Channel Bandwidth (CB); and the transmission time for each symbol based on the Guard Interval (GI). The expected duration is then computed with the following formula:

$$expected\ duration = DIFS + average\ backoff + data\ tx\ time + SIFS + ack\ tx\ time \quad (5.3)$$

Since the backoff time for a specific aggregated frame cannot be determined from the collected traces, the average backoff time ($7.5 \times Slot\ Time\ \mu s$) is used when computing *expected duration*.

We separate delays into two categories, WiFi, and the broadly defined non-WiFi, delay. While we compute both types of delays using Equation 5.1, we apply them differently depending on the delay type. The following sections characterize the two types of interference and show that they can be accurately simulated by the T-SIMn system.

5.1 WiFi Interference

WiFi interference consists of any WiFi traffic occurring on the same (or overlapping) channels used by the Access Point (AP) for the experiment. This traffic can be generated by: (1) a completely independent network with its own AP and clients; (2) other clients belonging to the same network as the experiment's AP; or (3) by the experiment's AP itself (e.g., beacon frames). Sources (1) and (2) behave similarly; they delay a sender's transmissions since their presence is detected during channel sensing, which causes the sender to backoff before re-attempting transmission. These sources increase the time, measured by the *rx cycles*, that the device spends receiving data on top of the Ack. Source (3) includes outbound WiFi transmissions, such as beacon frames and probe responses. These sources increase the time that the device spends transmitting data, measured by the *tx cycles*, on top of the data payload. In practice, we treat these three types of WiFi interference identically when computing delay during simulation using Equation 5.1. We now evaluate the accuracy of SIMn in simulating WiFi interference.

Experiment Setup:

We create a network between the AP (*sender*) and two desktop computer clients. One client acts as a *receiver*, while the other acts as a third-party sender (*WiFi-interferer*). For all experiments, the *WiFi-interferer* is set to a constant rate configuration of `1S-I4-SG-40M=90`. We choose this rate as it is not affected by path loss at the distance from both the *sender* and the *receiver*, so that it does not create an instance of a hidden terminal.

We perform the following steps to conduct constant rate configuration experiments (i.e., trace collection) for the rate configurations `1S-I4-LG-20M=39`, `1S-I4-SG-40M=90`, `2S-I4-SG-40M=180` and `3S-I4-SG-40M=270`:

1. Starting at time $t = 0$ seconds, the *sender* is set to the respective constant rate configuration.
2. At time $t = 25$, the *WiFi-interferer* begins transmitting to the *sender* at a rate of 3 Mbps for 25 seconds.
3. At $t = 50$, the interferer is stopped.
4. At $t = 75$, the *WiFi-interferer* begins transmitting to the *sender* at a rate of 10 Mbps for the remainder of the 100 second experiment.

We simulate constant rate experiments for each rate configuration using the collected trace as input to the simulator. We expect throughput to drop slightly after $t = 25$ when the *WiFi-interferer* begins transmitting, due to competition for access to the channel, and to return to baseline levels when the interferer is turned off. After $t = 75$, we expect throughput to drop significantly, as the *WiFi-interferer* is set to use more bandwidth than at $t = 25$ (though its Physical Layer Data Rate (PLDR) remains the same). Therefore, it will occupy the channel for a greater portion of time, decreasing throughput for the *sender*.

We plot throughput for each *simulated* constant rate experiment (referred to as simulated throughput), along with throughput for each *conducted* constant rate experiment (referred to as experimental throughput), in Figure 5.1. Each point at time t represents the mean throughput (with 95% confidence intervals) obtained during the preceding 5 seconds (i.e., in the interval $(t-5, t]$). For example, the point at $t = 5$ represents mean throughput during $t = 0$ to $t = 5$. Points for simulated throughput are artificially offset by 1 second to facilitate visual comparisons with the corresponding points for experimental throughput. We use these graphing properties for all time series plots in the thesis.

As shown in Figure 5.1, we find that the simulated throughput matches the experimental throughput for the rate configurations `1S-I4-LG-20M=39`, `1S-I4-SG-40M=90` and `2S-I4-SG-40M=180`. For the rate configuration `3S-I4-SG-40M=270`, simulated throughput matches the experimental throughput for the points at $t = 0$ to 75. However, we find that simulated throughput is lower than experimental throughput for the points at $t = 45$ to 50, and $t = 85$ to 100. Upon closer inspection, we find that error rates for `3S-I4-SG-40M=270` increase to over 4% when the interferer is transmitting, while they remain below 2% for

the other configurations. While we attempt to minimize error rate by avoiding a hidden terminal scenario, we cannot completely study WiFi delay in isolation from error rate, as the *sender* and *WiFi-interferer* may choose the same backoff duration and begin transmitting at the same time, resulting in packet loss and an increase in error rate. We specifically study simulation with error in Chapter 6.

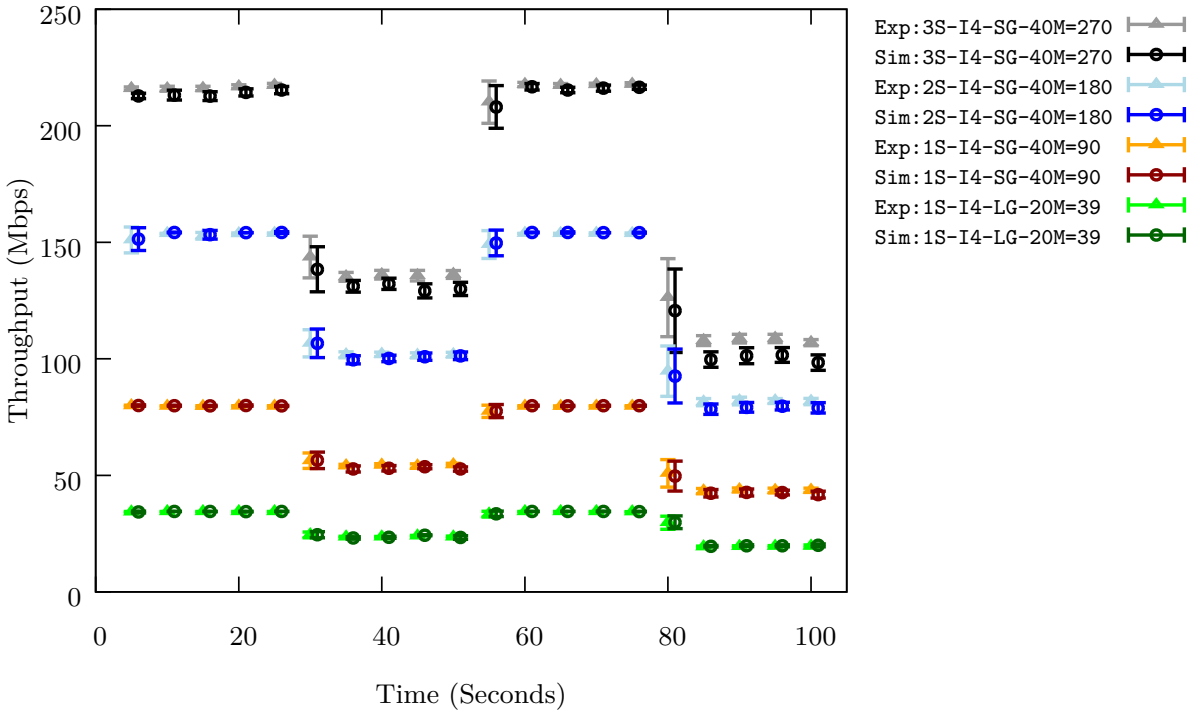

Figure 5.1: Simulated throughput when a WiFi interferer is turned on and off, compared against throughput measured experimentally.

In the previous experiment, we collect traces with $\texttt{FA}_{\texttt{COL}}$=MAX and simulate with $\texttt{FA}_{\texttt{SIM}}$=MAX, where the number of aggregated frames transmitted per unit of time is similar during collection and simulation. However, as described in Section 4.2.2, we often need to simulate $\texttt{FA}_{\texttt{SIM}} < \texttt{FA}_{\texttt{COL}}$. To handle this case, we must process WiFi delays differently from non-WiFi delays because they scale differently as $\texttt{FA}_{\texttt{SIM}}$ diverges from $\texttt{FA}_{\texttt{COL}}$. We illustrate this with an example: an AP is configured with a specific beacon interval (e.g., 50 ms); meaning that every 50 ms the AP must postpone transmission of data frames in order to transmit a beacon frame. If we simulate $\texttt{FA}_{\texttt{SIM}} < \texttt{FA}_{\texttt{COL}}$=MAX, we will send more (but shorter) aggregated frames per unit time than if we simulated $\texttt{FA}_{\texttt{SIM}}$=MAX. However, the number of

beacon frames transmitted in each window is independent of the number of aggregated frames (i.e., regardless of how many aggregated frames we transmit in a given period of time, a beacon frame will only be transmitted every 50 ms). If we compute a mean WiFi delay per aggregated frame from a trace collected with $\texttt{FA}_{\texttt{COL}}$=MAX, we will overestimate the total WiFi delay if we simulate $\texttt{FA}_{\texttt{SIM}} < \texttt{FA}_{\texttt{COL}}$ and apply the mean WiFi delay to each aggregated frame. The total delay would be increased by the ratio of aggregated frames transmitted during simulation compared to collection:

$$delay\ inflation = \frac{\#aggregated\ frames\ (simulation)}{\#aggregated\ frames\ (collection)} = \frac{\texttt{FA}_{\texttt{SIM}}}{\texttt{FA}_{\texttt{COL}}} \tag{5.4}$$

Rather than using the mean when computing WiFi delay, we apply WiFi delays obtained directly from the collected trace (e.g., the delay for a beacon is applied every 50 ms, rather than averaged over many frames). This relies on being able to distinguish WiFi from non-WiFi interference. To do this we use a heuristic proposed by Abedi [1], where a delay is considered to be from WiFi if one or both of the following are true:

1. The time spent transmitting the frame is significantly more than expected:

$$actual\ tx\ duration - expected\ tx\ duration > threshold$$

$$\frac{tx\_cycles}{clock\_speed} - data\ tx\ time > threshold$$

   This handles the case where the frame is delayed due to outbound traffic on the AP, such as transmitting a beacon frame or responding to a probe request. The heuristic uses a threshold of 60 μs as it is small enough to catch beacon frames, which were the shortest WiFi frames that we observed.

2. The time spent receiving the ack for the frame is significantly longer than expected:

$$actual\ rx\ duration - expected\ rx\ duration > threshold$$

$$\frac{rx\_cycles}{clock\_speed} - ack\ rx\ time > threshold$$

   This handles the case where a frame is delayed due to inbound traffic, such as receiving a data frame from another client. The heuristic uses a threshold of 10 μs because there is little variation in the *actual rx durations* in the absence of WiFi interference.

The thresholds above may be further tuned to improve the accuracy of the heuristic in differentiating WiFi and non-WiFi interference, though we find that these values provide accurate results in the following evaluation. We now evaluate the accuracy of simulating $\texttt{FA}_{\texttt{SIM}} < \texttt{FA}_{\texttt{COL}}$ using the heuristic to handle WiFi interference.

Experiment Setup:

We create a network between the AP (*sender*) and a desktop computer client (*receiver*). We collect a trace for the rate configuration `2S-I4-SG-40M=180` with $\texttt{FA}_{\texttt{COL}}$=`MAX`. We then conduct experiments with Frame Aggregation (FA) limited to 2 and 1 aggregated frames. Using the trace collected with $\texttt{FA}_{\texttt{COL}}$=`MAX`, we simulate constant rate scenarios for the rate configuration `2S-I4-SG-40M=180` with $\texttt{FA}_{\texttt{SIM}}$=`2` and $\texttt{FA}_{\texttt{SIM}}$=`1`. This is done to evaluate SIMn's accuracy in simulating shorter frames, from traces collected with $\texttt{FA}_{\texttt{COL}}$=`MAX`, in the presence of WiFi interference. We then repeat these simulations but disable the heuristic (i.e., WiFi and non-WiFi delays are not differentiated) to demonstrate the need to treat WiFi and non-WiFi interference differently.

In Figure 5.2, we plot the simulated throughput (with and without the heuristic) for $\texttt{FA}_{\texttt{SIM}}$=`2` and $\texttt{FA}_{\texttt{SIM}}$=`1`, along with the experimental throughput with FA limited to 2 and 1 aggregated frames. The bars for "Simulated (with Heuristic)" show throughput when simulating $\texttt{FA}_{\texttt{SIM}}$=`2` and $\texttt{FA}_{\texttt{SIM}}$=`1`, using the $\texttt{FA}_{\texttt{COL}}$=`MAX` trace as input to the simulator. The bars for "Simulated (without Heuristic)" also show throughput when simulating $\texttt{FA}_{\texttt{SIM}}$=`2` and $\texttt{FA}_{\texttt{SIM}}$=`1`, using the $\texttt{FA}_{\texttt{COL}}$=`MAX` trace as input to the simulator, but the heuristic for differentiating WiFi and non-WiFi interference is not used. For this experiment, the only WiFi interference is from beacon frames generated by the AP. We find that simulation error (i.e., the difference between simulated throughput, with and without the heuristic, and the experimental throughput), is lower when using the heuristic. With the heuristic, simulation error is less than 1% when simulating $\texttt{FA}_{\texttt{SIM}}$=`2` from $\texttt{FA}_{\texttt{COL}}$=`MAX`, and less than 2% when simulating $\texttt{FA}_{\texttt{SIM}}$=`1` from $\texttt{FA}_{\texttt{COL}}$=`MAX`. Without the heuristic, simulation error is roughly 6% and 10% for $\texttt{FA}_{\texttt{SIM}}$=`2` and $\texttt{FA}_{\texttt{SIM}}$=`1`, respectively. These are significant differences, especially when considering that the only WiFi interference is beacon frames, which are relatively short in duration compared to data frames. We expect these differences to be even larger if a third-party WiFi client is added.

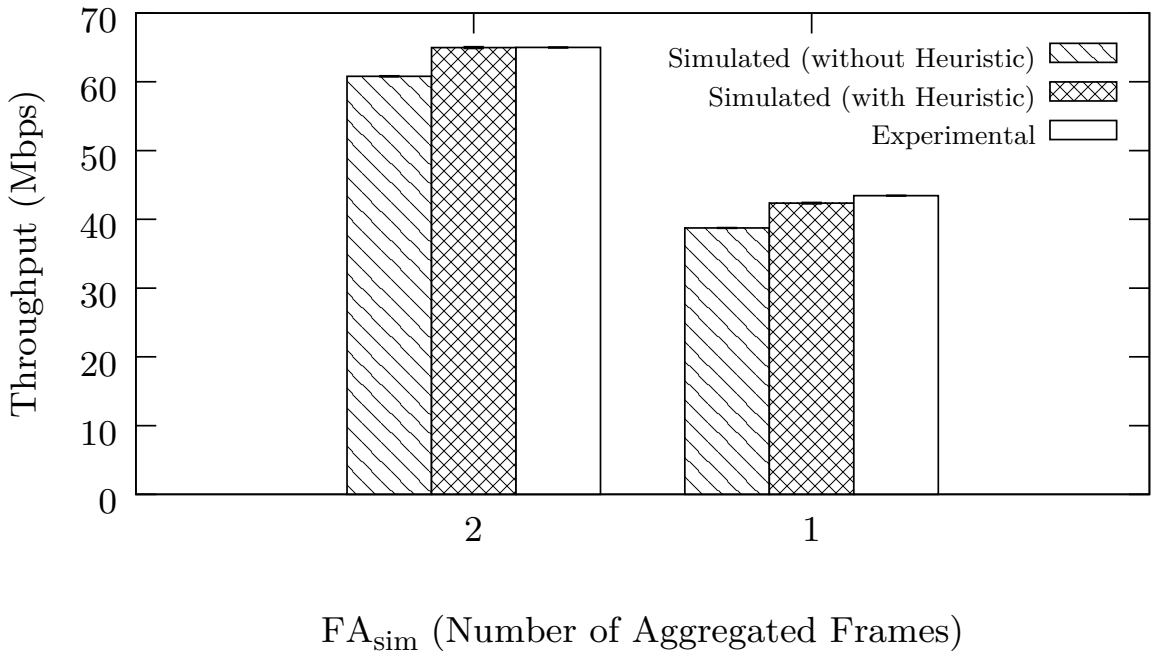

Figure 5.2: Throughput when aggregating fewer frames during simulation than collection.

5.2 Non-WiFi Interference

Non-WiFi interference consists of any energy detected on the channel and may be generated by many devices, some of which include: Bluetooth devices, microwave ovens, wireless keyboards/mice and cordless phones. It is common for these non-WiFi devices to use the Frequency-Hopping Spread Spectrum (FHSS) method for transmission and we focus on this class of devices in the evaluation of the simulator. For the purposes of simulation, we define non-WiFi interference as any interference that was not classified as WiFi using the heuristic presented above in Section 5.1. Non-WiFi interference results in increases in the *total_cycles* required to transmit an aggregated frame since the *sender* must backoff when it detects energy during channel sensing. Thus we again use Equation 5.1 for computing non-WiFi delay. In contrast to delays caused by WiFi interference, delay due to non-WiFi interference depends on the number of aggregated frames transmitted. Because each aggregated frame transmission will perform channel sensing, if we simulate a fixed rate

scenario where we send more individual frames per unit time, such as a scenario without frame aggregation, the mean delay will be increased by the ratio of aggregated frames transmitted during simulation compared to collection (Equation 5.4). The more frequently we perform channel sensing, the more likely we will be delayed. Note that this depends on the pattern of non-WiFi interference being received. For example, an interferer that only periodically transmits (e.g., once every second) may result in delays similar to those produced by WiFi devices. We now evaluate the accuracy of SIMn in simulating non-WiFi interference, similar to that generated by a cordless phone. We do not evaluate SIMn with different patterns of non-WiFi interference, though this could be an area for future research.

Experiment Setup:

We create a network between the AP (*sender*) and a desktop computer client (*receiver*). To generate non-WiFi interference we use the RF Signal Generator in frequency sweep mode over a range of 160 MHz. This mode is chosen to mimic the interference produced by an FHSS cordless phone. Since our AP has a CB of 40 MHz, the signal generator is broadcasting on a frequency overlapping with our AP roughly 1/4 of the time, when active. We perform the following steps to collect traces for the rate configurations `1S-I4-LG-20M=39`, `1S-I4-SG-40M=90`, `2S-I4-SG-40M=180` and `3S-I4-SG-40M=270`:

1. The *sender* is set to the respective constant rate configuration.
2. At time $t = 25$ *seconds*, the non-WiFi signal generator is enabled for 25 seconds.
3. At $t = 50$, the non-WiFi signal generator is disabled.
4. At $t = 75$, the non-WiFi signal generator is enabled for the remainder of the experiment.

We simulate constant rate experiments for each rate configuration using the collected trace as input to the simulator. We expect throughput to drop after $t = 25$ due to competition with the non-WiFi interferer for access to the channel. We then expect throughput to return to the levels observed in the first 25 seconds of the experiment, as the interferer is disabled. After $t = 75$, we expect throughput to drop to the same level as observed for the points $t = 30$ to $t = 50$ *seconds* because the same non-WiFi interference is generated until the experiment ends.

We plot the simulated throughput for each rate configuration, along with experimental (raw) throughput computed from the collected trace, in Figure 5.3. We find that the simulated throughput matches the experimental throughput for all four rate configurations, which suggests that non-WiFi interference is accurately simulated by SIMn.

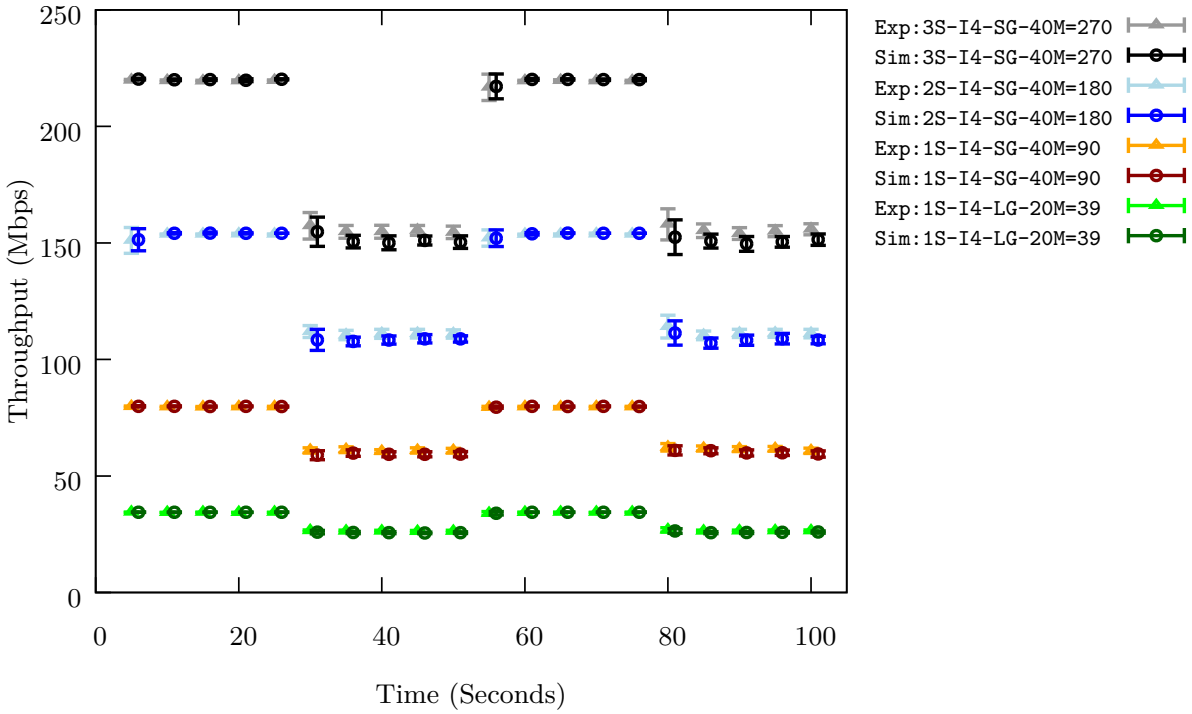

Figure 5.3: Simulated throughput when a non-WiFi interferer is turned on and off, compared against throughput measured experimentally.

5.3 Summary

We show that SIMn can accurately simulate delay due to both WiFi and non-WiFi interference. We also demonstrate that WiFi and non-WiFi interference must be handled differently in order to accurately simulate shorter aggregated frames when collecting with $\texttt{FA}_{\texttt{COL}}$=MAX (i.e., $\texttt{FA}_{\texttt{SIM}} < \texttt{FA}_{\texttt{COL}}$).

Chapter 6

Simulating Channel Error Rate

Along with the Physical Layer Data Rate (PLDR) and channel access (delay), channel error rate is one of the major factors in determining the throughput for an 802.11n network. Error can be introduced when multiple WiFi devices transmit at the same time, resulting in a packet collision (i.e., an undecodeable frame). Error can also be caused by non-WiFi interferers that begin transmitting after the device performs channel sensing and begins transmitting. Furthermore, error may be caused by path loss when signal strength is low due to the distance between a *sender* and *receiver* or because of obstacles like walls or furniture, that obscure the path between the two devices. We begin by describing the need to consider the subframe index when computing the fate of a subframe in Section 6.1 and then describe the simulation of path loss in Section 6.2.

6.1 Subframe Index Error Rates

An aggregated frame consists of multiple individual frames, which we refer to as subframes. Each subframe in an aggregated frame is indexed from 0 to $n-1$, with index 0 being transmitted first and $n-1$ last. In our initial implementation of SIMn, we treated the successes and failures of individual subframes as samples in the computation of an error rate for a time window. For example, if we sent 5 aggregated frames, each containing 30 subframes, 15 successful and 15 failing, we would compute the error rate to be 50% because a total of 75 subframes were successfully transmitted and 75 failed. We found that simulated throughput did not always closely match experimental throughput, even though the error rates obtained in the simulator were similar to those in the experiments. However, we found that the average length of aggregated frames differed significantly in

simulation compared to experimentation. We found that the error rate can vary with the subframe index. Byeon et al. find that that subframes transmitted more than 2 ms after the beginning of the transmission of an aggregated frame have a lower probability of being successfully received [8]. We find that the pattern of subframe error rates differs significantly across different scenarios. In some scenarios the pattern is similar to that found by Byeon et al., in others we see high error rates for the first subframe index with a sharp decrease in the following subframes indices, and in others we see subframe error rates that are roughly uniform across all subframe indices. Because we have observed several different patterns, we use a technique that will work with any pattern, including environments where the pattern changes over time. SIMn now computes individual error rates for each subframe index, rather than using an average error rate across the entire aggregated frame. We refer to this as a Subframe Index Error Rate (SFIER). We now evaluate the impact of SFIERs on simulated throughput.

Experiment Setup:

We generate two synthetic traces with equal overall error rates of 41%, but using two different SFIER patterns. We use synthetic traces due to the difficulty in experimentally obtaining traces with the same overall error rate with different SFIER patterns. The first trace has a linearly increasing SFIER from 0.025 at index 0 to 0.8 at index 31 (i.e., $SFIER = 0.025, 0.050, 0.075, \ldots, 0.775, 0.800$ for the indices $0, 1, 2, \ldots, 30, 31$, respectively). The second trace is the opposite, with the SFIERs decreasing linearly from 0.8 at index 0 to 0.025 at index 31 (i.e., $SFIER = 0.800, 0.775, 0.750, \ldots, 0.050, 0.025$ for the indices $0, 1, 2, \ldots, 30, 31$, respectively). We simulate throughput for the rate configuration `3S-I7-SG-40M=450` using the two synthetic traces, treating all subframe indices equally, as in our initial implementation of SIMn. We then repeat the simulations using our current implementation of SIMn which considers each SFIER individually. We expect that a decreasing SFIER pattern will result in lower throughput, as failures at the start of an aggregate frame will prevent the Block-Ack Window (BAW) (described in Section 4.2.1), from advancing and thus will reduce the average length of aggregated frames in the simulation.

We plot throughput for the two synthetic traces in Figure 6.1. The bars on the left show results obtained using the current implementation with per-SFIERs and those on the right show results obtained using the initial implementation where SFIERs are averaged. These results show that when considering per-SFIERs, an increasing SFIER pattern results in higher throughput than a decreasing pattern, even though they have the same average error rate. As expected, when all subframe indices are treated equally (i.e., they are averaged) we find that both patterns result in the same throughput. This is because the overall error

rate of both traces is the same, at 41%. These experiments demonstrate the importance of considering individual SFIERs.

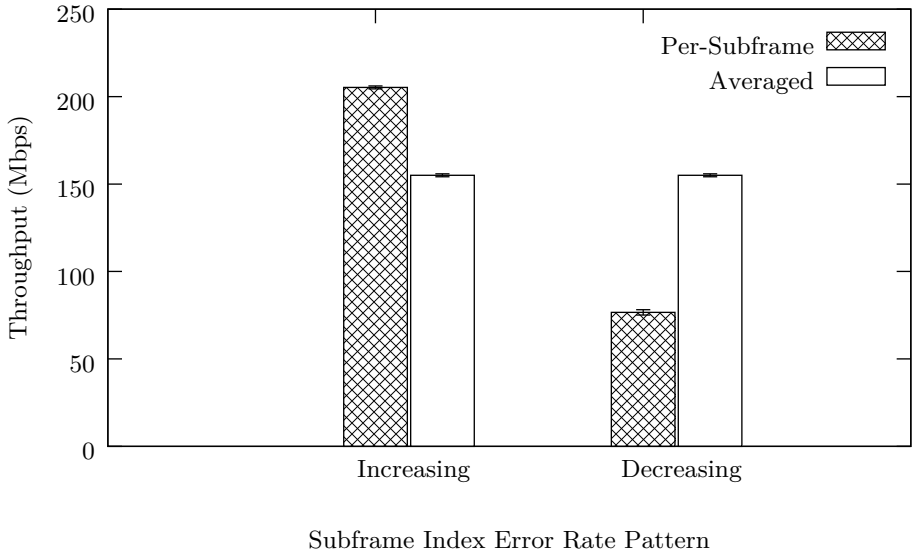

Figure 6.1: Simulated throughput for increasing and decreasing SFIERs when using per-subframe and averaged error rates.

6.2 Path Loss

To evaluate the simulation of channel error rate in T-SIMn we focus on path loss because, unlike WiFi and non-WiFi interference, it allows the channel error rate to be studied in isolation from channel access (delay). Due to Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA), WiFi devices will delay transmission in the presence of WiFi and non-WiFi interference to avoid transmitting at the same time as the interferer. Although collisions can still occur if the interferer begins transmitting after the WiFi device has begun transmitting, we find that error rates remain relatively low when using our non-WiFi signal generator to create interference. Therefore, we now evaluate the accuracy of SIMn using a challenging mobile scenario, with error rates that will be significantly impacted by path loss.

Experiment Setup:

We create a network between the Access Point (AP) (*sender*) and a MacBook Pro (MBP). The *sender* is set to a constant rate configuration of `2S-I4-SG-40M=180`. We choose this rate configuration because it experiences a wide range of error rates in this

mobile scenario. We collect a 100-second trace with `FA_COL=MAX`. At time $t = 0$ seconds, the MBP is held in close proximity (1 meter) to the *sender*. At $t = 10$, we begin walking slowly away from the *sender* in a C shape until reaching a wall at $t = 80$, roughly 10 m from the *sender* with cubicle walls obscuring line of sight. At $t = 85$, we quickly change direction, walking quickly towards the *sender* until trace collection ends at at $t = 100$. We expect that due to path loss, throughput will decrease as we move further away from the *sender* and that it will increase as we move towards the *sender*. Signal strength decreases exponentially with distance and is decreased by obstacles, such as walls, that block line of sight. Reduced signal strength leads to increased error rates and decreased throughput. We simulate throughput for the rate configuration `2S-I4-SG-40M=180`, using the collected trace as input to the simulator. This tests SIMn's ability to accurately simulate throughput with fluctuating error rates (with many samples due to constant rate trace collection).

In Figure 6.2, we plot pairs of throughput measurements, simulated and experimental, for the mobile scenario. This graph shows that throughput decreases quickly until $t = 20$ seconds. From $t = 20$ to $t = 80$, the throughput fluctuates significantly. At $t = 85$, there is a rapid drop in throughput as the MBP is moved quickly to change direction. Throughput then increases until the end of the trace, as we move closer to the *sender*. Despite the significant fluctuations from $t = 20$ to $t = 80$, there is a close match between simulated and experimental throughput in all time intervals. This suggests that the simulator can accurately simulate a variety of channel error rates.

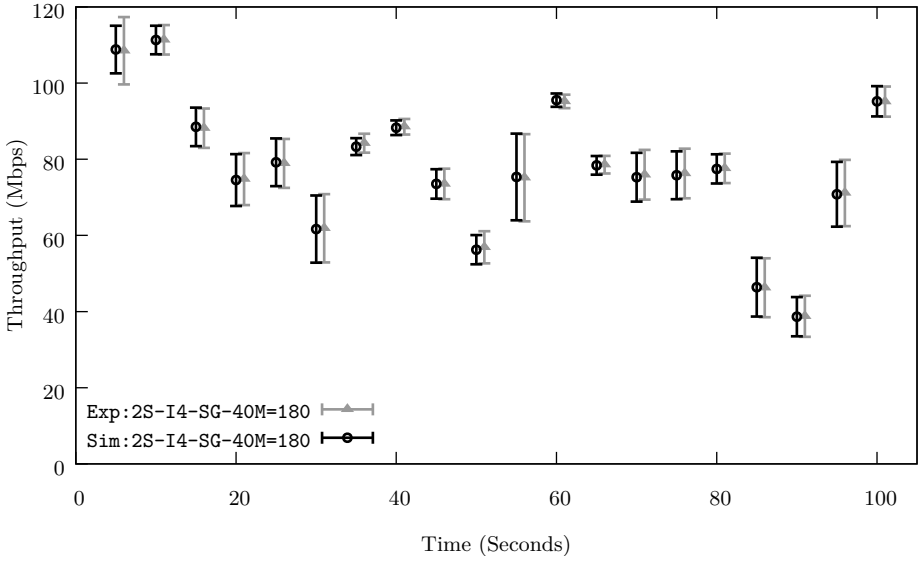

Figure 6.2: Simulated throughput for a mobile scenario experiencing path loss.

6.3 Summary

We show that in order to match the wide range of subframe loss patterns that can occur during trace collection, Subframe Index Error Rates (SFIERs) must be considered during simulation. We demonstrate that SIMn can accurately simulate a mobile scenario with a wide range of error rates. Unlike Chapter 4 and Chapter 5, we are unable to test the simulation of shorter aggregated frames due to a lack of experiment repeatability with path loss. However, the following chapter tests the entire framework in challenging scenarios that will require SIMn to simulate shorter aggregated frames from traces collected with $\texttt{FA}_{\texttt{COL}}$=MAX.

Chapter 7

Combining Trace Collection and Simulation

In the previous chapters, we use constant rate trace collection in order to test the T-SIMn simulator, SIMn. By limiting trace collection to constant rates, we have a large number of samples for each time interval. Although not representative of how trace collection would be done in T-SIMn, this technique is used to evaluate SIMn on its own. In this chapter, we evaluate the T-SIMn framework as a whole, using round-robin trace collection, in conditions that are representative of those in which WiFi is used. Recall from Chapter 6 that we are unable to obtain repeatable traces in scenarios that include path loss. Obtaining repeatable traces becomes even more challenging in uncontrolled environments, where interference from WiFi and non-WiFi devices is unpredictable. In order to evaluate T-SIMn in these conditions, we make use of an experimental methodology that is used to evaluate T-RATE, which does not require repeatability. We describe this approach in Section 7.1 and describe a revised evaluation methodology in Section 7.2 for 802.11n with Frame Aggregation (FA). We then evaluate T-SIMn in a completely uncontrolled environment, in Section 7.3.

7.1 Evaluating the T-SIMn Framework

To evaluate T-RATE in scenarios where repeatable traces could not be obtained, Abedi and Brecht [2] collect round-robin traces and use these traces to simulate a Rate Adaptation Algorithm (RAA) that should provide the same short and long term throughput, as is obtained when collecting the trace. This is done using a round-robin RAA but with a different

ordering from that used during trace collection. Using this RAA, each rate configuration is used for the same proportion of time during both trace collection and simulation. Therefore, the throughput that is obtained during trace during collection is expected to match simulated throughput. Because the round-robin ordering is different during trace collection and simulation, the fate of a packet transmitted at time t, with rate configuration r, cannot simply be obtained by looking up the entry in the collected trace for time t. This is done to validate the averaging window approach to determining a packet's fate, referred to as the trace preparation phase in T-RATE. We evaluate T-SIMn using the same methodology, first collecting traces by sampling all rate configurations in a round-robin fashion and then simulating a different algorithm for determining the rate configuration of each packet, in this case the algorithm is round-robin in a different order. During trace collection, rates are grouped by a combination of the Guard Interval (GI) and the Channel Bandwidth (CB). Rates are sampled in the following group order `LGI-20MHz`, `SGI-20MHz`, `LGI-40MHz`, `SGI-40MHz`. Within each group, rates are sampled in order from the lowest Modulation and Coding Scheme (MCS) to the highest (i.e., MCS $0, 1, ..., 6, 7$). We simulate round-robin in the reverse group order (i.e., `SGI-40MHz`, `LGI-40MHz`, `SGI-20MHz`, `LGI-20MHz`) and from the highest MCS to the lowest (i.e., MCS $7, 6, ..., 1, 0$). Recall from Section 3.3.1 that sampling all 96 rates supported by our WiFi devices takes over 300 ms, which is longer than the channel coherence time and averaging window that we use in T-SIMn. In order to evaluate T-SIMn in a real-world scenario, we limit trace collection to the set of 1-Spatial Stream (SS) rates. Widely popular small devices, such as cell phones, commonly have only one WiFi antenna due to power and space restrictions, and therefore only support 1-SS rates. In Section 8.3.1, we discuss the research topic of trace collection with 2 or more SSs, which is currently being explored. We now describe the experimental setup that is used to collect traces for evaluating T-SIMn.

Experiment Setup:

We create a network between the Access Point (AP) (*sender*) and an iPhone 6 (*receiver*). The sender is configured to sample all 1-SS 802.11n rate configurations. This includes 8 MCSs, 2 GIs and 2 CBs, for a total of 32 rate configurations. This is the entire set of 802.11n rate configurations supported by the iPhone 6. We collect two 100-second traces with $\texttt{FA}_{\texttt{COL}}\texttt{=MAX}$, each using a different scenario. In the first scenario, we test gradual increases and decreases in error rate, such as a person walking towards or away from an AP. We consider this scenario to be representative of typical mobile WiFi usage. In the second scenario, we test large increases in error rate, such as those experienced when a device is moved far away from an AP. This scenario represents very poor channel conditions, where a device may be close to losing connectivity with the AP. We now describe the experimental methodology for collecting traces for the two scenarios:

Scenario 1

At time $t = 0$ *seconds*, the *receiver* is held in close proximity (1 meter) to the *sender*. We begin walking slowly away from the *sender* for 55 seconds until we reach a wall. Between $t = 55$ and $t = 65$, we slowly turn around. We then begin walking slowly towards the *sender* until trace collection ends at $t = 100$. The *receiver*'s positions at $t = 0$ and $t = 100$ are approximately the same (i.e., the *receiver* returns to the starting position at the end of trace collection).

Scenario 2

At time $t = 0$ *seconds*, the *receiver* is held further from the *sender* than in Scenario 1, at roughly 4 meters away, facing a door that allows us to exit the room quickly. The *receiver* remains stationary for the first 20 seconds of the experiment. At $t = 20$, the door is opened and we walk through, allowing the door to close behind us. The path between the *sender* and the *receiver* is now obstructed by a wall and a door made of metal, which lowers the signal strength. We walk further away from the *sender*, stopping at $t = 50$. We remain stationary for 20 seconds, holding the phone still. At $t = 70$, we walk towards the door, moving closer to the *sender*. At $t = 80$, we open the door and quickly enter the room, allowing the door to close behind us. At $t = 90$ s, we turn around to face the door, returning to the starting position. We remain still until trace collection ends at $t = 100$.

In both scenarios, we expect that throughput will decrease as we move further from the *sender* and increase as we move closer. We expect throughput to be lower in the second scenario due to the *receiver* being further from the *sender* throughout the entire experiment and due to path loss from the wall and door.

In Figure 7.1 and Figure 7.2, we plot simulated and experimental throughput for the two scenarios, respectively. In Scenario 1, we find that simulated throughput matches experimental throughput (i.e., they have overlapping confidence intervals) except for the points at time $t = 20$ to 25, where simulated throughput is roughly 5% lower than experimental throughput. In Scenario 2, we find that simulated throughput matches experimental throughput except for the points at times $t = 20$, $t = 60$ and $t = 100$. The largest difference is observed at $t = 60$, where simulated throughput is roughly 11% higher than experimental throughput. At times $t = 20$ and $t = 100$, simulated throughput is roughly 1% lower than experimental throughput. While there is a throughput match for the majority of the time intervals in both scenarios, in terms of overlapping confidence intervals, the match is not as close as with T-RATE for 802.11g networks. Initially, we thought that this was due to inaccuracy in SIMn but upon closer investigation, we realized that the opposite is true. Our assumption that simulating round-robin in a different order would

result in each rate configuration being used for the same proportion of time, in both trace collection and simulation, is no longer guaranteed in 802.11n networks. In the following section, we investigate and explain why the sampling order of round-robin trace collection impacts throughput.

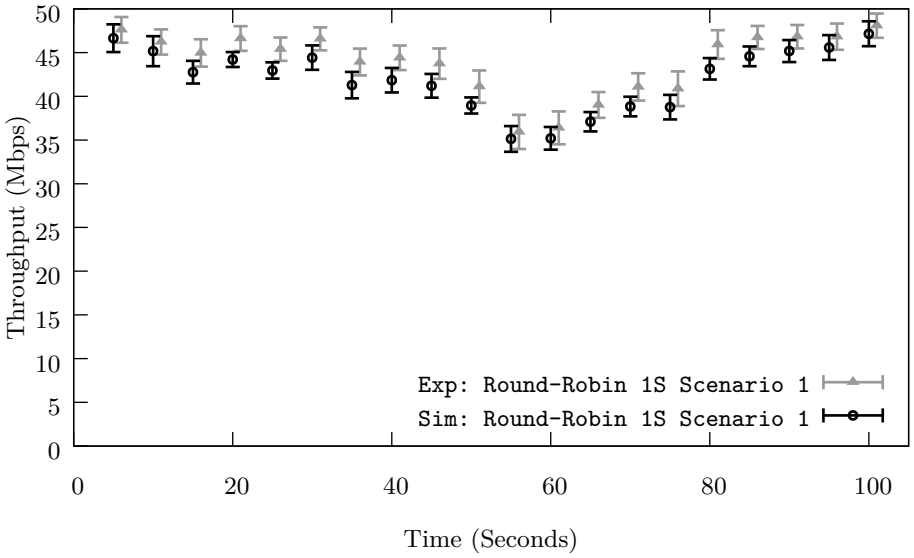

Figure 7.1: **Scenario 1** Round-robin simulated in the reverse order of trace collection.

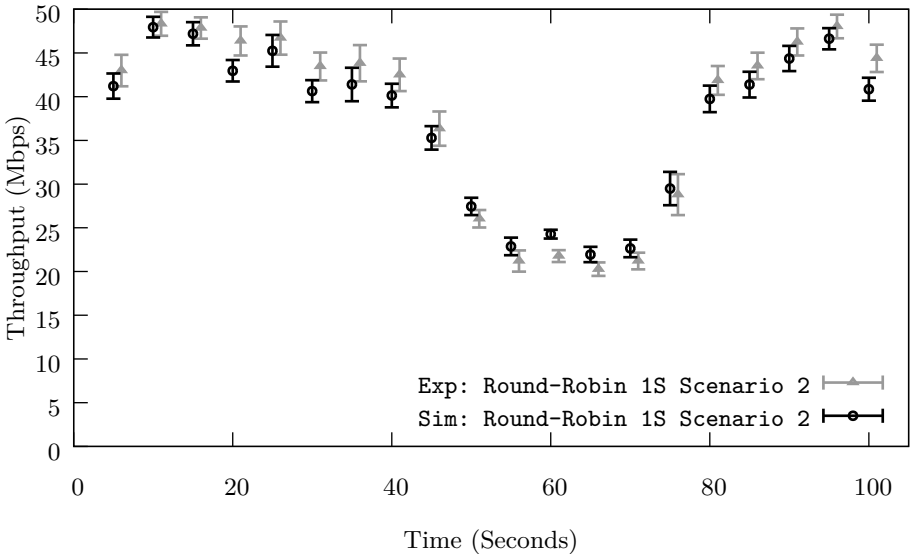

Figure 7.2: **Scenario 2** Round-robin simulated in the reverse order of trace collection.

7.2 The Importance of Rate Configuration Ordering

When using round-robin trace collection with T-RATE for 802.11g networks, each rate configuration that is sampled is independent from the last (i.e., the fate of the previous frame does not impact the fate of the current frame). This allows round-robin trace collection to be compared against a round-robin simulation performed in any order. However, in 802.11n networks, the fate of previous frames impacts Frame Aggregation (FA) due to the Block-Ack Window (BAW). Failed aggregated frames, or subframes, limit how far forward the BAW can be advanced. This can, in turn, limit the length of the current aggregated frame. Recall from Figure 1.1 that the number of subframes being aggregated has a significant impact on throughput, with longer aggregated frames leading to higher potential throughput. In T-SIMn, we sample rates in the group order `LGI-20MHz`, `SGI-20MHz`, `LGI-40MHz`, `SGI-40MHz`. Importantly, the rates are sampled in order from the lowest Modulation and Coding Scheme (MCS) to the highest (i.e., MCS $0, 1, ..., 6, 7$). This means that the most robust rates in each group are sampled first and the least robust rates are sampled last. In Scenario 2, shown in Figure 7.2, we have examined the data in detail and found that at times $t = 50$ to 70, the reverse ordering performed during simulation leads to longer aggregated frames on average (a mean of 15.2 subframes in each aggregated frame during simulation compared to 14.6 in the experiment), which results in slightly higher throughput during this period. In contrast, the average length of aggregated frames was higher in the remaining time intervals of the trace, with aggregated frames being shorter on average during simulation than experimentation. Although there is a match in simulated and experimental throughput during these time intervals (i.e., overlapping confidence intervals), the simulated throughputs are visibly lower for most times t. Simulating longer frames than those that were collected may also lead to inaccuracy due to a lack of statistics for subframes with higher indices than those in the collected frames.

We now expect different round-robin orderings to result in different throughput, unless the behavior of the Block-Ack Window (BAW) advancement and consequently Frame Aggregation (FA), is the same during trace collection and simulation. To test this hypothesis, we construct a new ordering that preserves the property that the most robust rates in each group are sampled first and the least robust rates are sampled last. We sample rate groups in the reverse order, `SGI-40MHz`, `LGI-40MHz`, `SGI-20MHz`, `LGI-20MHz`. However, within each group, we sample rates in order from the lowest Modulation and Coding Scheme (MCS) to the highest (i.e., the same as during trace collection). We simulate round-robin in this new reverse group order and show simulated and experimental throughput for Scenario 1 and Scenario 2, in Figure 7.3 and Figure 7.4, respectively. We now observe a very close match of simulated and experimental throughput in both scenarios, for all time intervals, with

confidence intervals overlapping in all cases. Note that this property does not limit SIMn to simulating only certain orderings of rate configurations. It is only when evaluating the accuracy of T-SIMn by comparing with results obtained from an experiment that we must consider this property. Now that we are aware of this property, we will use the reverse group ordering in the following section, where we evaluate T-SIMn in an uncontrolled environment. In Section 8.3, we discuss possible trace collection techniques to avoid aggregated frame length being limited by the BAW, to avoid the potential inaccuracy that we mention above.

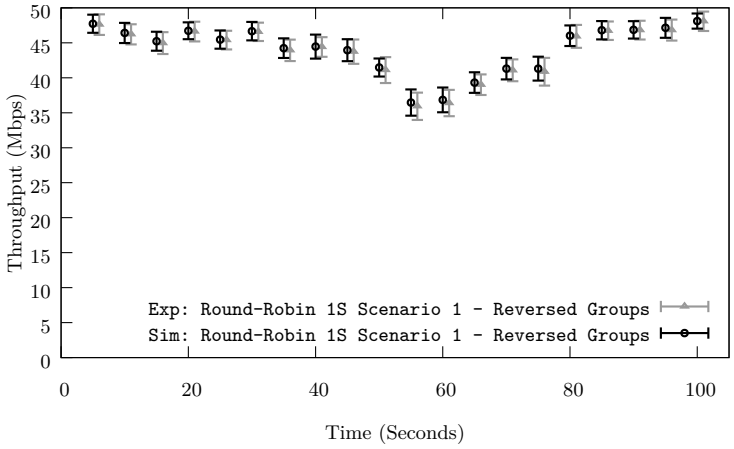

Figure 7.3: **Scenario 1** Round-robin simulated in the reverse rate configuration group order of the trace (i.e., `SG-40M`, `LG-40M`, `SG-20M`, `LG-20M`), but MCS indices going from low to high within each group (i.e., $0, 1, ..., 7$), like the trace.

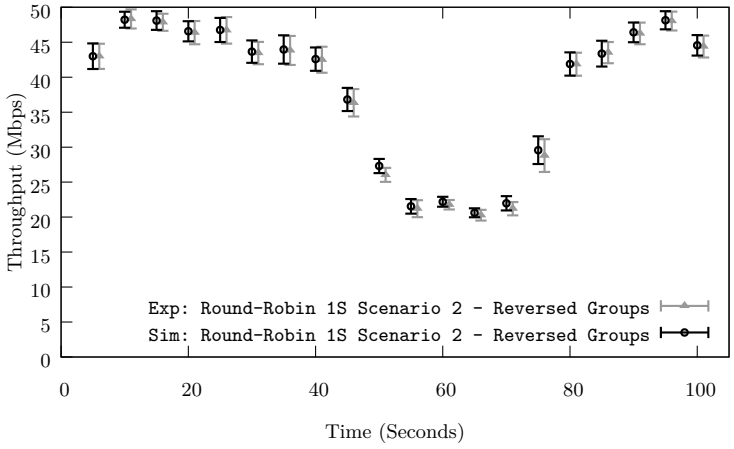

Figure 7.4: **Scenario 2** Round-robin simulated in the reverse rate configuration group order of the trace (i.e., `SG-40M`, `LG-40M`, `SG-20M`, `LG-20M`), but MCS indices going from low to high within each group (i.e., $0, 1, ..., 7$), like the trace.

7.3 Uncontrolled Trace Collection and Simulation

Up to this point, all experiments are performed on a 5 GHz channel with no neighboring Access Points (APs). We now move to a different 5 GHz channel that is in use by the university's WiFi network, to evaluate T-SIMn in conditions that are typical for a shared university WiFi network. This includes interference from many third-party WiFi clients and APs. We now describe the setup that is used to collect traces in this environment.

Experiment Setup:

Similarly to the previous section, we create a network between the AP (*sender*) and an iPhone 6 (*receiver*). However, unlike previous experiments, we now configure the AP to use a channel occupied by one of the university's APs. We choose the 5 GHz channel that is occupied by the AP with the highest signal strength. We use a 5 GHz channel because the iPhone does not support 40 MHz Channel Bandwidths (CBs) in the 2.4 GHz spectrum. Note that if we had used the 2.4 GHz band, thus limiting trace collection to 20 MHz rate configurations, we would have obtained twice as many samples in each averaging window which should only improve accuracy. As in previous experiments, we sample all 1-Spatial Stream (SS) 802.11n rate configurations. We collect a 100 second trace with $\texttt{FA}_{\texttt{COL}}\texttt{=MAX}$ to test T-SIMn in an uncontrolled environment. At time $t = 0$ *seconds*, the *receiver* is held in close proximity (1 meter) to the *sender*. At $t = 25$, we walk to the metal door, leaving the room at $t = 35$. At $t = 45$, we re-enter the room and stop moving at $t = 60$. We remain stationary for the remainder of trace collection, ending at $t = 100$. We check for matches between simulated and experimental throughput for this trace.

In Figure 7.5, we plot pairs of throughput measurements, simulated and experimental, for the uncontrolled experiment. We find that while the *receiver* is stationary from $t = 60$ to 100 in Figure 7.5, there is significantly more fluctuation in throughput when compared to Scenario 2 from $t = 10$ to 40 in Figure 7.4. This is due to delay from third-party WiFi traffic on the shared channel. This is expected due to students using the network. The close matches in throughput suggest that T-SIMn is accurately capturing and simulating third-party traffic and that T-SIMn can accurately simulate conditions that are representative of those in which WiFi devices are used.

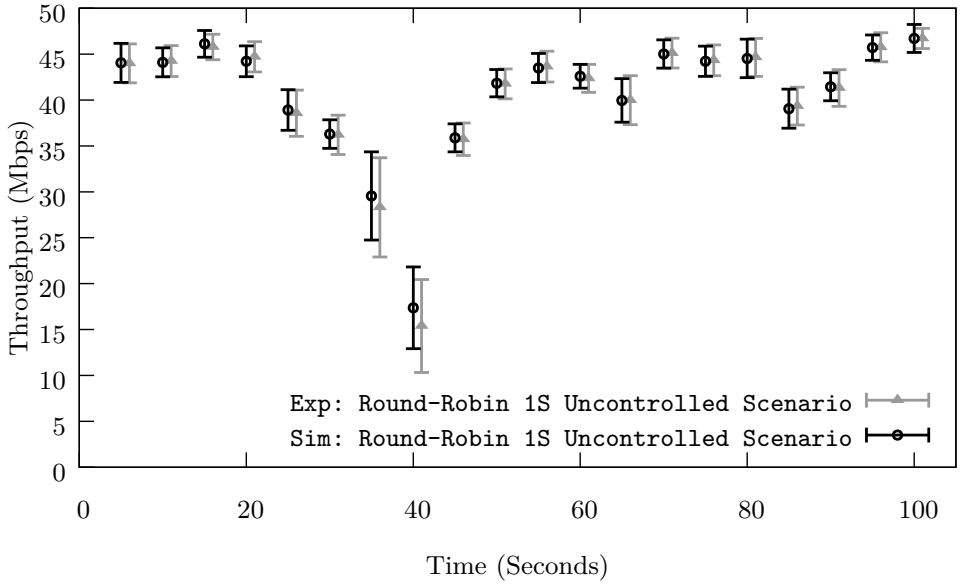

Figure 7.5: **Uncontrolled Mobile Scenario** Simulated round-robin throughput for a mobile scenario experiencing path loss and WiFi interference. Round-robin is simulated in the reverse rate configuration group order of the trace (i.e., `SG-40M`, `LG-40M`, `SG-20M`, `LG-20M`), but MCS indices going from low to high within each group (i.e., $0, 1, ..., 7$), like the trace.

7.4 Summary

We collect round-robin traces and simulate round-robin using a different ordering of rate configurations to evaluate the T-SIMn framework in conditions that are representative of those in which WiFi devices are used. However, initially the match between experimental and simulated throughput was not as close as was obtained with T-RATE for 802.11g. We then investigated the evaluation methodology in greater detail and discovered that the round-robin ordering of rate configurations impacts throughput due to Frame Aggregation (FA) in 802.11n, which was unexpected. This demonstrates the power of T-SIMn because this property would be challenging to observe experimentally. Using the traces and SIMn we were able to understand this important property. We correct the evaluation methodology and obtain a close match in simulated and experimental throughput which suggests that the T-SIMn framework is highly accurate in simulating 802.11n devices with 1 antenna in uncontrolled environments.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

In this thesis, we describe the development of a trace-based simulation framework for 802.11n networks, called T-SIMn, that is based on an existing framework, T-RATE. In particular, we focus on the implementation of the simulator component of the framework, called SIMn. We find that careful consideration of both the physical layer and MAC layer features of 802.11n is necessary in order to accurately simulate this standard. We demonstrate that SIMn accurately simulates one, two and three-antenna rates in 802.11n with Frame Aggregation (FA), by obtaining throughput metrics from the simulator that closely match those obtained experimentally. We show that aggregated frames of any length (i.e., $\texttt{FA}_{\texttt{SIM}}$=1, $\texttt{FA}_{\texttt{SIM}}$=2, ..., $\texttt{FA}_{\texttt{SIM}}$=MAX can be accurately simulated from longer aggregated frames (i.e., $\texttt{FA}_{\texttt{COL}}$=MAX). Furthermore, we find that SIMn accurately simulates delay due to WiFi and non-WiFi interference, and error rates due to path loss. Despite the complexity of the interdependent physical and MAC layer features in 802.11n, we are able to accurately simulate challenging mobile scenarios using SIMn.

We develop a new implementation for the trace collection phase and demonstrate that in its current state, the T-SIMn framework can be used with 1 Spatial Stream (SS) (i.e., 1 antenna) devices. The iPhone 6 (and all earlier versions) contain one WiFi antenna, along with most phones from other manufacturers. As a minor indication of the number of devices this framework can be applied to, over 230 million iPhones were sold in Apple's 2015 fiscal year alone [6].

With so many devices being limited to 1 SS, T-SIMn is a valuable tool for mobile phones. The groundwork has also been laid for simulating more SSs (i.e., support for these rates has

been implemented and tested in SIMn). Full trace-based simulations should be possible once better trace collection techniques are developed. We discuss the limitations of T-SIMn in greater detail in Section 8.2 and discuss avenues for future research or improvements to T-SIMn in Section 8.3.

8.2 Limitations

While we show that in its current state SIMn can simulate 802.11n networks with WiFi interference, non-WiFi interference and path loss, we envision that it will become a platform for fairly comparing competing algorithms (e.g., Rate Adaptation Algorithms (RAAs) or Frame Aggregation Algorithms (FAAs)) or configurations (e.g., Long Guard Intervals (LGIs) and Short Guard Intervals (SGIs), or 20 and 40 MHz Channel Bandwidths (CBs)) for 802.11n networks. We now discuss the limitations of the current implementation of T-SIMn and limitations of the evaluation for SIMn.

8.2.1 Tight Coupling with Channel Coherence Time

To determine the fate of a frame at time t, T-SIMn uses samples that are collected within the channel coherence time to compute an error rate. Samples falling outside the channel coherence time may be collected in channel conditions that differ from those at time t. Because of this requirement, T-SIMn's approach to determining the fate of a frame is tightly coupled with the channel coherence time. We limit trace collection to 1 antenna rate configurations in the evaluation of T-SIMn in order to obtain sufficient samples for each rate configuration in the channel coherence time. We also limit the evaluation to scenarios that are stationary or use walking speeds, as channel coherence time generally decreases as movement speeds increase. Further evaluation is required to better understand how different environments and different movement speeds affect channel coherence times, and to understand the impact of channel coherence times on the ability of the T-SIMn framework to identically match the physical reality of a scenario. Note, however, that we are referring to the ability of T-SIMn to capture channel conditions that exactly match reality, rather than the accuracy in simulating different algorithms. Although almost all scenarios examined in this thesis have very close matches, we may not be able to capture channel conditions as accurately in environments with shorter coherence times. It is important to understand that the framework still allows for the fair comparison of different algorithms. Each algorithm is subjected to the exact same channel conditions, even if the channel conditions do not identically match reality.

8.2.2 Evaluations Requiring Repeatability

To evaluate the accuracy of T-SIMn in simulating shorter aggregated frames from traces collected with $\texttt{FA}_{\texttt{COL}}$=MAX we require experimental repeatability. Recall that we use an experiment run with $\texttt{FA}_{\texttt{COL}}$=N as ground truth and compare against a simulation of $\texttt{FA}_{\texttt{SIM}}$=N from the trace collected with $\texttt{FA}_{\texttt{COL}}$=MAX. Environmental conditions must be the same during the collection of the trace and all experiments that we wish to conduct (i.e., conditions must remain the same during the collection of the trace with $\texttt{FA}_{\texttt{COL}}$=MAX and the experiments with $\texttt{FA}_{\texttt{COL}}$=N for all values of N being tested). In the evaluation of T-RATE [2], repeatbility was obtained by conducting experiments in an environment with no external interference and using an electric train to move the sender at the same speed, following the same path in each run of an experiment. Unfortunately, we were unsuccessful in creating scenarios with experimental repeatability that include path loss or WiFi and non-WiFi interference (i.e., we found that throughput differed between identical runs of experiments using these scenarios) in 802.11n with FA. Our evaluation of simulating shorter aggregated frames is therefore limited to error-free scenarios, in which we are able to obtain consistent throughput between runs of each experiment, and indirectly in the uncontrolled scenarios in Chapter 7, where simulated aggregated frame lengths do not always match those in the experiments.

We would also require experimental repeatability in order to evaluate T-SIMn with real RAAs, as we would need to compare simulated throughput from a round-robin trace against an experiment using the RAA. To avoid this requirement, we evaluate T-SIMn using an RAA that we can expect to produce the same throughput as obtained during trace collection. In this case we simulate round-robin in a different ordering. For further evaluation of T-SIMn and SIMn we hope to construct repeatable experiments using the controlled introduction of errors, possibly using different frequency hopping patterns than were used in Section 5.2 with the signal generator, controlling the signal strength of the sender, or both.

8.3 Future Work

We now discuss future areas of research to address the limitations described in the previous section and specific use cases for T-SIMn that we hope to explore.

8.3.1 Trace Collection with More Rate Configurations

In Chapter 7, we have evaluated T-SIMn when sampling at most 32 rate configurations, by limiting the set of rates to those that use 1-SS only. However, 802.11n networks support 128 rate configurations through a combination of 4 SSs, 8 Modulation and Coding Schemes (MCSs), 2 Guard Intervals (GIs) and 2 CBs. Sampling the 96 rate configurations supported by our devices (maximum of 3 SSs) in a round-robin fashion requires over 300 ms, which means that each round is longer than the channel coherence time of 100 ms that we use in T-SIMn. As a result, we are not able to obtain enough samples to closely match simulated and experimental throughput when collecting with all 96 rate configurations. This problem is being investigated by Ali Abedi, the co-author of T-RATE [2], using correlations between rate configurations. Early results are promising and we expect this to allow a greater number of rates to be sampled in T-SIMn in the future, making it useful for devices with more than 1 antenna.

8.3.2 Simulating Rate Adaptation Algorithms

We expect that T-SIMn will be capable of simulating statistically-based 802.11n RAAs, which rely on error rate statistics to choose which rate configurations to use, similarly to how T-RATE simulates RAAs for 802.11g. We provide an RAA interface in SIMn that is similar to the *mac80211* module in the Linux kernel. This should allow common RAAs, such as Minstrel-HT, to be ported to the simulator. However, validating the T-SIMn simulator with RAAs will be challenging due to the difficulty in obtaining repeatable traces with path loss. While the current approach to trace collection relies on error rate statistics, it may be possible to collect Channel State Information (CSI) [14], which provides additional link quality information, to evaluate a wider range of RAAs. For example, the Effective SNR [13] RAA uses CSI to predict the highest rate configuration that will be successfully received, rather than using statistics from past frames to make predictions.

8.3.3 Better Evaluation of Frame Aggregation Algorithms

In Section 4.2.2, we show that T-SIMn is capable of accurately simulating shorter aggregated frames, despite performing trace collection with $\texttt{FA}_{\texttt{COL}}$=MAX. However, this evaluation is performed in an error-free environment. We show that simulated and experimental throughput matches in the presence of interference (Chapter 5) and path loss (Chapter 6). Although we do not directly study aggregation of shorter frames from $\texttt{FA}_{\texttt{COL}}$=MAX traces in

these environments, we have shown that in uncontrolled environments, the T-SIMn framework is highly accurate. Further research in this area may allow FAAs to be evaluated in T-SIMn. Byeon et al. [8] find that decreasing the maximum length of aggregated frames can increase throughput in a mobile environment due to their observation that higher subframe indices had higher error rates. They propose an FAA, called MoFA [8], which adapts the length of aggregated frames based on the Subframe Index Error Rates (SFIERs). Since the T-SIMn framework uses SFIERs, it should be an ideal platform for evaluating and comparing existing and new FAAs. Much like with RAAs, T-SIMn could allow multiple FAAs to be fairly compared using common traces, ensuring fair comparisons and reducing the experimental effort required. For example, MoFA is evaluated by comparing average throughput of multiple mobile trials, with and without the FAA. This evaluation methodology is time consuming and may be unfair, as it is challenging to repeat mobile experiments (e.g., maintaining the exact same walking speed and path across trials).

8.3.4 Simulating 802.11ac

802.11ac is the successor to 802.11n and was officially ratified in January 2014. 802.11ac supports even more rates than 802.11n, with the availability of 80 and 160 MHz CBs (in addition to the existing 20 and 40 MHz CBs), as well as up to 8 SSs (compared to 4 SSs in 802.11n). MAC layer features such as FA and Block Acknowledgment (BA) remain similar in 802.11ac, which may make the transition from 802.11n to 802.11ac simpler than from 802.11g to 802.11n. Unfortunately, much of the functionality in Ath9k has been moved to closed source firmware in the Ath10k 802.11ac device driver, which may make it challenging to port T-SIMn to 802.11ac using the current drivers and hardware.

8.4 Concluding Remarks

We believe that T-SIMn is a valuable tool for studying widely used devices with 1 antenna and that work currently in progress will enable complex, multi-antenna devices to be studied in the future. We look forward to T-SIMn being used to fairly and easily evaluate new RAAs and FAAs in environments that are representative of those in which WiFi devices are typically used.

APPENDICES

Appendix A

802.11 Rate Tables

For reference, Table A.1 and Table A.2 show the Physical Layer Data Rates (PLDRs) in 802.11g and 802.11n, respectively. These tables also show the combinations of modulation types and coding rates (as well as Guard Intervals (GIs) and Channel Bandwidths (CBs) in 802.11n) that lead to each PLDR.

Table A.1: 802.11g Rate Table [12]

MCS Index	Modulation Type	Coding Rate	Data Rate (Mbps)
0	BPSK	1/2	6
1	BPSK	3/4	9
2	QPSK	1/2	12
3	QPSK	3/4	18
4	16-QAM	1/2	24
5	16-QAM	3/4	36
6	64-QAM	2/3	48
7	64-QAM	3/4	54

Table A.2: 802.11n Rate Table [12]

				Data Rate (Mbps)			
				20 MHz		40 MHz	
# SSs	MCS Index	Modulation Type	Coding Rate	LGI	SGI	LGI	SGI
1	0	BPSK	1/2	6.5	7.2	13.5	15.0
1	1	QPSK	1/2	13.0	14.4	27.0	30.0
1	2	QPSK	3/4	19.5	21.7	40.5	45.0
1	3	16-QAM	1/2	26.0	28.9	54.0	60.0
1	4	16-QAM	3/4	39.0	43.3	81.0	90.0
1	5	64-QAM	2/3	52.0	57.8	108.0	120.0
1	6	64-QAM	3/4	58.5	65.0	121.5	135.0
1	7	64-QAM	5/6	65.0	72.2	135.0	150.0
2	0	BPSK	1/2	13.0	14.4	27.0	30.0
2	1	QPSK	1/2	26.0	28.9	54.0	60.0
2	2	QPSK	3/4	39.0	43.3	81.0	90.0
2	3	16-QAM	1/2	52.0	57.8	108.0	120.0
2	4	16-QAM	3/4	78.0	86.7	162.0	180.0
2	5	64-QAM	2/3	104.0	115.6	216.0	240.0
2	6	64-QAM	3/4	117.0	130.0	243.0	270.0
2	7	64-QAM	5/6	130.0	144.4	270.0	300.0
3	0	BPSK	1/2	19.5	21.7	40.5	45.0
3	1	QPSK	1/2	39.0	43.3	81.0	90.0
3	2	QPSK	3/4	58.5	65.0	121.5	135.0
3	3	16-QAM	1/2	78.0	86.7	162.0	180.0
3	4	16-QAM	3/4	117.0	130.0	243.0	270.0
3	5	64-QAM	2/3	156.0	173.3	324.0	360.0
3	6	64-QAM	3/4	175.5	195.0	364.5	405.0
3	7	64-QAM	5/6	195.0	216.7	405.0	450

References

[1] Ali Abedi. Personal communication, 2015.

[2] Ali Abedi and Tim Brecht. T-RATE: A framework for the trace-driven evaluation of 802.11 rate adaptation algorithms. In *Modelling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2014 IEEE 22nd International Symposium on*, pages 1–10. IEEE, 2014.

[3] Ali Abedi, Andrew Heard, and Tim Brecht. Conducting repeatable experiments and fair comparisons using 802.11n MIMO networks. *ACM SIGOPS Operating Systems Review*, 49(1):41–50, 2015.

[4] AirMagnet. Fluke networks. http://www.flukenetworks.com/enterprise-network/-wireless-network/AirMedic.

[5] Jeffrey G Andrews, Arunabha Ghosh, and Rias Muhamed. *Fundamentals of WiMAX: understanding broadband wireless networking*. Pearson Education, 2007.

[6] Apple Inc. Annual Report - Form 10-K, 2015. http://investor.apple.com/secfiling-.cfm?filingID=1193125-15-356351.

[7] Ryan Burchfield, Ehsan Nourbakhsh, Jeff Dix, Kunal Sahu, S Venkatesan, and Ravi Prakash. RF in the jungle: Effect of environment assumptions on wireless experiment repeatability. In *IEEE International Conference on Communications*, pages 1–6. IEEE, 2009.

[8] Seongho Byeon, Kangjin Yoon, Okhwan Lee, Sunghyun Choi, Woonsun Cho, and Seungseok Oh. MoFA: Mobility-aware frame aggregation in Wi-Fi. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, pages 41–52. ACM, 2014.

[9] Lara Deek, Eduard Garcia-Villegas, Elizabeth Belding, Sung-Ju Lee, and Kevin Almeroth. Joint rate and channel width adaptation for 802.11 MIMO wireless networks. In *10th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 167–175. IEEE, 2013.

[10] Lara Deek, Eduard Garcia-Villegas, Elizabeth Belding, Sung-Ju Lee, and Kevin Almeroth. Intelligent channel bonding in 802.11n WLANs. *IEEE Transactions on Mobile Computing*, 13(6):1242–1255, 2014.

[11] Lara Deek, Eduard Garcia-Villegas, Elizabeth Belding, Sung-Ju Lee, and Kevin Almeroth. A practical framework for 802.11 MIMO rate adaptation. *Computer Networks*, 2015.

[12] Mattew S. Gast. *802.11n: A Survival Guide*. O'Reilly, 2012.

[13] Daniel Halperin, Wenjun Hu, Anmol Sheth, and David Wetherall. Predictable 802.11 packet delivery from wireless channel measurements. *ACM SIGCOMM Computer Communication Review*, 41(4):159–170, 2011.

[14] Daniel Halperin, Wenjun Hu, Anmol Sheth, and David Wetherall. Tool release: Gathering 802.11n traces with channel state information. *ACM SIGCOMM Computer Communication Review*, 41(1):53–53, 2011.

[15] Christopher Hepner, Arthur Witt, and Roland Muenzner. In depth analysis of the ns-3 physical layer abstraction for WLAN systems and evaluation of its influences on network simulation results. *International Workshop on Socially Intelligent Computing*, pages 46–51, 2015.

[16] Pei Huang, Xi Yang, and Li Xiao. Adaptive channel bonding in multicarrier wireless networks. In *Proceedings of the fourteenth ACM international symposium on Mobile ad hoc networking and computing*, pages 297–300. ACM, 2013.

[17] Glenn Judd and Peter Steenkiste. Repeatable and realistic wireless experimentation through physical emulation. *ACM SIGCOMM Computer Communication Review*, 34(1):63–68, 2004.

[18] Glenn Judd and Peter Steenkiste. Using emulation to understand and improve wireless networks and applications. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, Volume 2*, pages 203–216. USENIX Association, 2005.

[19] Glenn Judd and Peter Steenkiste. Characterizing 802.11 wireless link behavior. *Wireless Networks*, 16(1):167–182, 2010.

[20] Glenn Judd, Xiaohui Wang, Mei-Hsuan Lu, and Peter Steenkiste. Using physical layer emulation to optimize and evaluate mobile and wireless systems. In *Proceedings of the 5th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*, Mobiquitous '08, pages 26:1–26:10. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[21] Lito Kriara and Mahesh K Marina. SampleLite: A hybrid approach to 802.11n link adaptation. *ACM SIGCOMM Computer Communication Review*, 45(2):4–13, 2015.

[22] Mathieu Lacage and Thomas R Henderson. Yet another network simulator. In *Proceeding from the 2006 Workshop on ns-2: The IP Network Simulator*. ACM, 2006.

[23] Feng Lu, Geoffrey M Voelker, and Alex C Snoeren. SloMo: Downclocking WiFi communication. In *NSDI*, pages 255–258, 2013.

[24] Jouni Malinen. hostapd: IEEE 802.11 AP, IEEE 802.1X, 2015. https://w1.fi/-hostapd/.

[25] Peter Matulis. Centralised logging with rsyslog. *Canonical Technical White Paper*, 2009.

[26] Nuts About Nets. RF Explorer Handheld, RF Signal Generator, 2015. http://-rfexplorer.com/rf-signal-generator/.

[27] ns-3 project. ns-3 Wi-Fi module design documentation, 2016. https://www.nsnam-.org/docs/models/html/wifi-design.html.

[28] Ioannis Pefkianakis, Yun Hu, Starsky H.Y. Wong, Hao Yang, and Songwu Lu. MIMO rate adaptation in 802.11n wireless networks. In *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking*, MobiCom '10, pages 257–268. ACM, 2010.

[29] Ioannis Pefkianakis, Suk-Bok Lee, and Songwu Lu. Towards MIMO-aware 802.11n rate adaptation. *IEEE/ACM Transactions on Networking*, 21(3):692–705, 2013.

[30] SCALABLE Network Technologies, Inc. QualNet, 2014. http://web.scalable-networks.com/content/qualnet.

[31] Wei-Liang Shen, Kate Ching-Ju Lin, Shyamnath Gollakota, and Ming-Syan Chen. Rate adaptation for 802.11 multiuser MIMO networks. *IEEE Transactions on Mobile Computing*, 13(1):35–47, 2014.

[32] ESnet Software. iperf3, 2015. http://software.es.net/iperf/.

[33] Michiel Steyaert and Willy MC Sansen. *Design of Multi-Bit Delta-Sigma A/D Converters*. Springer Science & Business Media, 2002.

[34] Mirko Stoffers and George Riley. Comparing the ns-3 propagation models. In *IEEE 20th International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 61–67. IEEE, 2012.

[35] Riverbed Technology. SteelCentral Riverbed Modeler, 2016. http://www.riverbed-.com/products/steelcentral/steelcentral-riverbed-modeler.html.

[36] Peyman Teymoori, Aresh Dadlani, Khosrow Sohraby, and Kiseon Kim. An optimal packet aggregation scheme in delay-constrained IEEE 802.11n WLANs. In *8th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, pages 1–4. IEEE, 2012.

[37] Ajay Tirumala, Feng Qin, Jon Dugan, Jim Ferguson, and Kevin Gibbs. Iperf: The TCP/UDP bandwidth measurement tool, 2005. http://sourceforge.net/projects/-iperf2/.

[38] Federico Tramarin, Stefano Vitturi, and Michele Luvisotto. Improved rate adaptation strategies for real-time industrial IEEE 802.11n WLANs. In *IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, pages 1–8. IEEE, 2015.

[39] Zenghua Zhao, Fucheng Zhang, Shaoping Guo, Xiang-Yang Li, and Junze Han. RainbowRate: MIMO rate adaptation in 802.11n WiLD links. In *IEEE International Performance Computing and Communications Conference (IPCCC)*, pages 1–8. IEEE, 2014.

[40] Pei Zheng and Lionel M. Ni. EMWIN: Emulating a mobile wireless network using a wired network. In *Proceedings of the 5th ACM International Workshop on Wireless Mobile Multimedia*, WOWMOM '02, pages 64–71. ACM, 2002.

[41] Junlan Zhou, Zhengrong Ji, and Rajive Bagrodia. TWINE: A hybrid emulation testbed for wireless networks and applications. In *INFOCOM, Volume 6*, pages 23–29, 2006.