IEEE 22nd International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, Sept., 2014 (MASCOTS 2014)

T-RATE: A Framework for the Trace-Driven Evaluation of 802.11 Rate Adaptation Algorithms

Ali Abedi and Tim Brecht
Cheriton School of Computer Science, University of Waterloo
ali.abedi@uwaterloo.ca | brecht@cs.uwaterloo.ca

Abstract

Rate adaptation algorithms (RAAs) in 802.11 networks can have a significant impact on network throughput. In order to attempt to maximize throughput, these algorithms dynamically adapt to inferred changes in the channel being used. Evaluating and comparing rate adaptation algorithms is extremely challenging due to the variability of channel conditions. Recent work on developing and evaluating new RAAs has used traces to increase the realism of simulators.

We propose a new framework for the trace-driven evaluation of RAAs (called T-RATE) in which we collect and process traces in environments where interference is caused by WiFi and non-WiFi devices. T-RATE enables a more comprehensive evaluation of RAAs in environments that are representative of those in which 802.11 devices are actually used. A key to our approach is that we capture, in traces, information that is relevant to RAAs regarding how channel conditions affect channel access and channel error rates. Our approach minimizes the use of wireless channel models and achieves highly realistic results under a wider variety of scenarios than previously possible. Our evaluation of T-RATE demonstrates it can be used to collect and process traces to accurately evaluate a variety of RAAs under more diverse and representative channel conditions than previously possible.

I. Introduction

The throughput of a device using a WiFi network depends on its rate adaptation algorithm's ability to successfully infer the quality of the communication channel being used and to select the rate that maximizes the device's bandwidth. Experimental evaluation and comparison of RAAs is extremely challenging due to the variability of channel conditions. Non-WiFi devices, such as microwave ovens, cordless phones, and other devices operating in the same spectrum make the situation worse. On the other hand, the complexity of wireless channels makes the accurate simulation or emulation of RAAs extremely difficult (and possibly infeasible).

As a result, recent evaluations of RAAs use traces to increase the realism of simulators and emulators [20] [13] [22] [11] [9]. Traces are collected and used to capture *some* aspects of wireless channel conditions, such as frame loss due to path-loss and fading, in order to avoid having to simulating/emulating them. Because the current approaches to collecting traces are able to capture only some aspects of wireless channel conditions, the environments under which this approach can be used is currently limited. For instance, the most recent approach [20] is limited to interference-free environments for trace collection, making it difficult to use in some important scenarios that are representative of those in which devices are likely to be used.

Throughput is the main performance metric used to evaluate 802.11 RAAs. It can be affected by the sender's ability to *access* the channel and the *error rate* observed when using the channel. Figure 1 depicts the environmental factors that affect the throughput. The properties of a channel that have been used in previous trace-driven evaluations of RAAs are represented by the shaded area. In contrast, our work is driven by the insight that data regarding the important properties of a wireless channel that affect the throughput and comparison of 802.11 RAAs can be captured in traces collected during real experiments. In other words, we capture and utilize all of the information regarding the factors shown in Figure 1. As a result, we are able to simulate the operation of the 802.11 MAC layer (as it relates to RAAs) to precisely mimic the properties of the wireless channel that affect throughput. The result is a framework for the fair and accurate comparison of RAAs using scenarios that are more representative of actual use conditions than previously possible. The contributions of this paper are:

- We collect, in traces, environmental factors that affect the throughput of RAAs. This includes information pertaining to Channel Access (carrier sensing and virtual carrier sensing) and Channel Error Rate (signal propagation, WiFi and non-WiFi interference).
- We have implemented a trace-processing engine that relies on relatively simple and yet highly accurate channel models. This enables the fair comparison and evaluation of multiple rate adaptation algorithms using identical channel conditions.
- We evaluate the efficacy of a trace-driven approach for comparing RAAs in 802.11 networks and show that it can produce highly accurate results.

Our prototype implementation of T-RATE uses 2.4 GHz, 802.11g networks. Our goal is to first determine if T-RATE will work for 802.11g networks before studying more complex 802.11n networks.

II. Related Work

To our knowledge, the first proposal for a trace-based evaluation of a wireless network appeared in 1997 [12]. Delay and loss statistics, computed from a trace of wireless transmissions, are used to train a wireless channel model. This approach was applied to a WaveLAN network with only a single transmission rate. Unfortunately, modern networks such as IEEE 802.11

support multiple transmission rates which makes collecting traces significantly more challenging (see Section III-A).

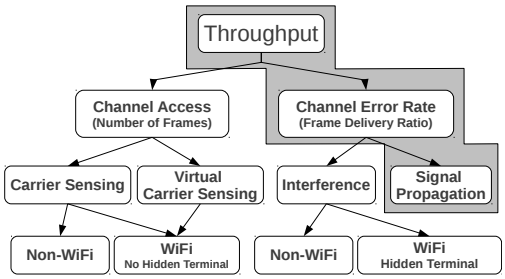


Fig. 1: Factors affecting throughput

More recently, traces have been used to evaluate RAAs by combining results obtained from experiments with a simulator or an emulator in an attempt to increase the realism of their channel models. Table I illustrates (with a check mark) the information collected in a trace by those papers that have used traces to compare RAAs. Note that in Table I, we compare the performance evaluation methodologies used in the referenced papers and *not the RAAs*. In some previous work [22] [9], some portions of the physical models that are commonly used in simulators or emulators are replaced with the traces collected in real experiments. In these cases a trace containing SNR (Signal-to-Noise Ratio) [22] or CSI (Channel State Information) [6] values obtained from an actual experiment is used to improve the realism of a simulator (SNR-to-SIM) and an emulator (SNR-to-EMU) [9]. In SNR-to-SIM and SNR-to-EMU, only SNR/CSI information is collected in a trace and frame loss is simulated or emulated from this data. Unfortunately, the SNR often inaccurately predicts the fate of a packet [6] [4]. Although the CSI provides a more accurate measure of channel quality than the SNR [6], both are oblivious to WiFi and non-WiFi interference and do not reflect the level of interference experienced at the receiver. Therefore, accurate frame loss estimation using SNR/CSI can be very challenging in presence of interference.

Another approach that utilizes traces replaces the channel propagation model in the NS-3 simulator with an actual trace which specifies the fate of a frame (i.e., frame loss) at any given time (FL-to-SIM) [20] [13] [19]. This approach is more realistic than SNR-to-SIM and SNR-to-EMU approaches, because it directly measures frame loss in contrast with estimating frame loss from SNR. In this case, traces are collected in *ideal interference-free environments* and packet loss due to path loss and fading is stored in the traces. Interference free environments are required by this approach because the trace collection mechanism is unable to completely capture the effect of interference. An additional problem is that wireless channels are quite complex and cannot be modeled accurately using only path loss information. For instance, the channel error rate can also be affected by WiFi and non-WiFi interference. In contrast, T-RATE does not require an interference-free environment for trace collection and can be used to collect traces to be used to evaluate RAAs using scenarios that are more representative of environments in which they will actually be used (see Section III-A).

As shown in Table I, previous work focuses only on capturing channel propagation properties in a trace (the shaded area in Figure 1). Ignoring any environmental factor during trace collection introduces inaccuracies in the results obtained using these traces. Therefore, the environmental factors that affect throughput must be captured to accurately replicate real environments. None of the work shown in Table I is able to collect traces in uncontrolled environments, instead they simulate the effect of environment factors. In contrast, T-RATE captures these factors intrinsically in traces (CT[1]-to-SIM) collected during actual experiments, thus simplifying and minimizing the use of wireless channel models.

Previous work ignores, simulates or emulates carrier sensing and virtual carrier sensing. Unfortunately, modeling carrier sensing is as complex as modeling frame loss because it involves modeling signal propagation for interfering sources. It is extremely difficult to estimate the effect of a WiFi or non-WiFi source on the channel observed at the sender, since signal propagation from all other sources should be modeled and requires complex wireless channel models. As a result, simple channel models that suffer from a lack of realism are used. Similarly, previous work ignores or simulates frame loss due to interference. WiFi interference (the hidden terminal problem) is simulated using simple and unrealistic models. For instance, it is assumed that any overlap of two frames results in the loss of both frames [20]. However, in reality, one of these frame may be received correctly if the signal strength of these frames differ considerably.

To our knowledge, none of these method are able to handle non-WiFi interference[2]. This is another reason why the environments in which these techniques can be used is restricted. Recent studies show that non-WiFi devices are widely used, including public places, and in enterprise and residential environments [3] [14]. A surprisingly high level of activity for non-WiFi devices is reported even at midnight [14]. Hence, a trace collection methodology that can be used to evaluate RAAs in such environments needs to capture the effect of non-WiFi devices. By intrinsically capturing environmental factors that affect throughput, shown in Figure 1, we achieve two important goals: 1) We minimize the use of wireless channel models that reduce realism. 2) Our framework can be used in a wider variety of environments than previously possible.

III. Trace-driven Framework

At a high-level, the goal of our approach is to enable traces to be collected with relative ease, under a variety of channel conditions. To evaluate the performance of rate adaptation algorithms, each trace is processed using the algorithms being studied. Because each algorithm uses exactly the same trace and each trace captures the channel conditions relevant to rate adaptation algorithms, this enables the fair, realistic, and repeatable comparison of algorithms. Figure 2 provides an overview of our proposed framework. First, in the trace collection phase, a trace of 802.11 transmissions are collected to capture different environmental factors that affect the throughput. Only a single frame with certain properties (e.g., 54 Mbps transmission rate) can be collected at time T. However, when we later simulate a rate adaptation algorithm,

[1]Compete Trace.

[2]Emulators can incorporate non-WiFi interference for some devices such as cordless phones that can be physically attached to the emulator but they cannot handle many other types of devices, microwaves for instance.

Methodology	Frame-loss (propagation)	Frame-loss (interference)	SNR	(V) CS (WiFi)	CS (non-WiFi)
SNR-to-SIM [22] [6]	✗ Simulated	✗ N/A	✓ Data frames	✗ Simulated	✗ N/A
SNR-to-EMU [9]	✗ Emulated	✗ Emulated	✓ Data frames	✗ Emulated	✗ Emulated
FL-to-SIM [20] [19] [16]	✓ Receiver-based	✗ Sim. WiFi interference	✗ N/A	✗ Simulated	✗ N/A
T-RATE: CT-to-SIM	✓ Sender-based	✓ WiFi & non-WiFi	✓ Data and ACK	✓	✓

TABLE I: Comparison of trace-based performance evaluation methodologies – (V) CS: (Virtual) Carrier Sensing

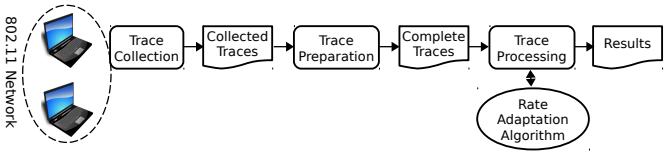


Fig. 2: Overview of our Trace-driven Framework

a frame with different properties (e.g., 48 Mbps transmission rate) might be required at time T. This frame is *missing* in the collected trace. Therefore, in the trace preparation phase, we estimate missing frames using collected frames and store them in complete traces. Finally, our trace-processing engine simulates the operation of the 802.11 MAC layer, UDP, and RAAs using the complete trace to report performance results.

A. Trace Collection

The goal of the trace-collection phase is to obtain a trace of an actual experiment whose 802.11 frame transmissions can be used to generate a representative trace for use by the trace-processing engine to evaluate RAAs. During this phase, an 802.11 network, consisting of an access point and a client (sender) is built. The sender saturates the link by transmitting as many frames as possible to the access point. We now describe the techniques we have developed to capture the environmental factors shown in Figure 1.

1) Collecting channel access data: All 802.11 devices perform carrier sensing before the transmission of a frame. The delay caused by a busy channel lowers the effective throughput. We use two mechanisms to detect and capture the effect of WiFi and non-WiFi interference on channel access.

WiFi sources: We use TCPDUMP [18] to capture the traffic generated from a sending device in addition to traffic that does not belong to our experiment (third party traffic). Third party traffic is used by the trace-processing engine to mimic the delay caused by detecting WiFi transmissions during carrier sensing.

Non-WiFi sources:[3] Capturing the effect of undecodable signals from non-WiFi devices detected during carrier sensing (energy detection) is more challenging. We exploit some information reported by widely used Atheros-based chipsets to calculate and capture the delay caused by energy detection on the channel. This information has been previously used for localization [5] and channel utilization estimation [1]. To our knowledge, we are the first to use this information to estimate the amount of time a channel is busy due to non-WiFi interference. The Atheros-based chipsets can report Cycle Counter Information (CCI) that indicates the time the chip spends transmitting and receiving data. This information is critical to obtaining accurate timing for packet transmission so we now describe the details of how it is obtained and used.

We have modified the Ath9k driver to report CCI information at the end of every transmission (i.e., when an ACK is received or is timed out). The chip uses a 40 MHz clock and in addition to the total number of cycles between transmissions (cycles), it reports the number of cycles spent receiving a frame (rx_frame), spent transmitting a frame (tx_frame), and the number of cycles during which the channel was busy (rx_busy).

In an interference-free environment, we expect to have: rx_busy = tx_frame + rx_frame. However, when the sender detects undecodable energy on the channel during carrier sensing (e.g., non-WiFi interference), we have: rx_busy $>$ tx_frame + rx_frame, because in addition to the transmission of the frame and possibly receiving the ACK, the channel was busy due to non-WiFi interference. Moreover, non-WiFi interference causes additional delays, since the channel needs to remain idle for the duration of a DCF inter frame space (DIFS) before the back-off count down can be resumed. In general, data transmission is delayed by one DIFS and possibly a portion of a back-off time slot for each detected burst of energy. Figure 3 shows two different examples where the same amount of busy time can result in significantly different delays before being able to transmit data. The example in the top line shows data transmission being delayed by three DIFS because two bursts of energy were detected. While the bottom line shows that despite energy being detected for the same amount of time, the packet is delayed by only two DIFS because only one burst of energy was detected.

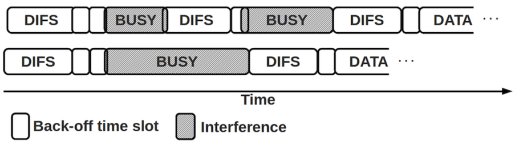


Fig. 3: Interference and channel sensing

Energy bursts may also occur during DIFS which can lead to loosing partial DIFS. Therefore, we used the following formula to calculate the total delay caused by interference detected at sender:

$$Delay = \frac{cycles - (tx\_frame + rx\_frame)}{Clock\ Frequency} - (DIFS + BO \times BO\_time\_slot + SIFS)$$

where BO is the back-off value used (i.e., the number of back-off time slots), and BO_time_slot is the time for each back-off. The first line of the formula (the fraction) calculates the

[3]In addition to non-WiFi devices such as cordless phones and microwaves, we treat WiFi devices operating on overlapping channels as non-WiFi interferers, because like non-WiFi sources their signal can not be decoded.

total time not sending or receiving WiFi frames. The second line deducts the standard wait times defined by the 802.11 standard (even when there is no interference). As a result, $Delay$ will represent any additional delay caused by non-WiFi interference. Because BO is chosen by the physical layer and is not reported to other layers, it is not possible to determine the value used. Therefore, we use an average back-off value of 7.5 in this equation, because values are chosen uniformly between 0 and 15. During the evaluation of our prototype implementation, we found that it was necessary to include CCI along with this heuristic to accurately capture the effect of non-WiFi interference on the sender.

2) Collecting channel error rate data: A key challenge of trace collection is that in order to later process a trace, we would like to be able to determine if a packet could be successfully received if it was sent at any available rate r at time t for all R available rates. However, while collecting a trace, we can send a packet using only one of the R supported rates, at any given time t. Unfortunately, sending packets from multiple devices on the same sender would causes packet collisions at the receiver.

Vutukuru et al. [20] propose sampling each of the available R rates in a strict round-robin fashion to get a sample from each rate r in a short time window W. The trace file records whether or not the frame transmission was successful for each sampled rate. Later during simulation the fate of the single packet sampled at rate r in a particular window W_i is used to determine the fate of all packets that could be sent at rate r in the same time window W_i. One disadvantage of this approach is that during simulation it assumes the frame error rate is either 0% or 100% for all sampling windows. In some cases, tens of frames can be transmitted during a time window. Unfortunately, some rate adaptation algorithms are highly sensitive to the consecutive failure or success of frames and may react unrealistically if evaluated using this technique. Therefore, we use a larger time window (i.e., 100 ms) to obtain several samples for each rate. We then compute the expected error rate for each rate during each time window. We choose a time window that is smaller than the expected *channel coherence time*. The channel coherence time is the time over which the channel conditions remain relatively constant and the channel can be assumed to be stationary. For 802.11 networks operating in the 2.4 GHz spectrum, where mobility is limited to a walking pace[4], studies show that the channel coherence time is between 100 and 200 ms [15], [17]. Sadeghi et al. [15] report that the channel coherence time reduces to 24 ms, 12 ms, and 6 ms for speeds of 18, 36, and 72 km/hr, respectively. In our prototype implementation, we use a 100 ms time window (50 ms before and after the point in time being considered). Hence, we require the coherence time to be above 50 ms which is the case when the speed of a mobile device is limited to walking speed. Our evaluations show that the design choice of 100 ms for the sampling window produces accurate results for experiments with mobility speeds up to walking speed. Studying higher speeds such as those in vehicular networks are left for future work.

We use a different technique to scan rates than used previously [20]. During the course of our investigation, we observed that traces collected using a strict round-robin strategy were not as realistic as desired. We observed that the length of each round can be synchronized with the duty cycle[5] of some non-WiFi interferers such as frequency-hopping spread spectrum (FHSS) devices like cordless phones. These devices transmit signals periodically at a particular frequency. Hence, some frames sent at a particular rate can experience interference with higher probability than others. This bias in error rate for some rates affects the realism of the results obtained from traces collected using a strict round-robin strategy. In some experiments, we observed that a strict round-robin strategy measures error rates with a difference as large as 21% when compared with base measurements. The base measurements were obtained using actual experiments that use a constant rate. By comparison, our pseudo-random approach obtained differences of less than 4.8%.

To our knowledge, a novel technique we use is the addition of scanning RTS/CTS protected frames. Previous work [20] performs sampling in interference free environments, and therefore ignores RTS/CTS. However, our work targets environments where WiFi interference may exist. In this case, the frame error rate depends on the use of the RTS/CTS protection mechanism. Our goal is also to support RAAs that may selectively enable and disable RTS/CTS [21]. As a results, we scan every other round with RTS/CTS option enabled.

In order to increase the realism of collected traces, we considered shortening the time required to complete a round by sampling only a subset of all available rates. For example, consider two simple heuristics for reducing the number of samples. 1) Sample rates in order from higher rates to lower rates. If k consecutive rates are successful, assume all rates lower than the lowest sampled rate will be successful (because they use more robust encodings). 2) Sample rates in order from lower rates to higher rates. If k consecutive rates are unsuccessful, assume all rates higher than the highest sampled rate will also be unsuccessful.

When analyzing data collected in environments where there is no interference the accuracy of these techniques is relatively high (even when a device is mobile). Unfortunately, such heuristics do not work in general due to the presence of interference which is surprisingly prevalent in environments in which WiFi devices are used [3] [14].

To study the problem, we collected and examined several traces collected by sampling all rates and then post processing the trace to determine whether or not the heuristics could accurately determine the fate of the packets that would not be sampled using the heuristic. Table II shows that the fate of a significant percentage of packets would be incorrectly predicted. As a result of these experiments, our prototype samples all rates. This provides highly accurate results in the scenarios we have examined (see Section V).

k	High to Low	Low to High
2	14%	64%
3	12%	52%

TABLE II: Errors in packet fate estimation heuristics

[4]The channel coherence time is inversely proportional to frequency, therefore the channel coherence time of networks operating in the 5 GHz is lower (about 100 ms).

[5]The fraction of time a device occupies the spectrum by transmitting signals when the device is active.

If, in future work, we encounter scenarios where we are not able to sample all rates or need to increase the accuracy of T-RATE, we would consider using software defined radios. For example, we could shorten rounds, as is done in AccuRate [16], by extracting physical layer information about the constellation of a received frame to estimate the fate of other rates that use different modulation and coding schemes.

3) The trace data: All information is collected on the sending device, except for the SNR of the received frame (which is collected on the receiver). In addition to channel access and channel error rate data, we collect other information, such as the SNR of data frames and acknowledgments, to enable the trace processing engine to evaluate RAAs that require this information (like RBAR [7] and CHARM [10]). Note that, if available, addition information could be collected. For example, physical layer properties could be obtained via special purpose hardware or from software defined radios [20].

The *collected traces* (Figure 2) are the Experiment Traffic Trace (ETT) and the Third-party WiFi Traffic Trace (TWTT). The ETT includes all frames transmitted by the sender and the per frame delay to access the channel caused by non-WiFi sources. The data in the ETT includes:

- Source: Ath9k Driver.
  - Channel Cycle Information (CCI).
- Source: TCPDUMP sender side.
  - Timestamp: the transmission time of the packet.
  - 802.11 transmission rate.
  - Was the packet sent using RTS/CTS.
  - ACK: specifies if the packet was received.
  - SNR of ACK.
- Source: TCPDUMP receiver side.
  - SNR of Data frame.

The ETT includes channel access data from non-WiFi but not WiFi sources. Therefore, the TWTT collects frames from interfering WiFi traffic that cause additional channel access delays at the sender. The data in the TWTT includes:

- Source: TCPDUMP sender side.
  - Timestamp: arrival time of the frame.
  - Frame length and transmission rate: used to calculate the time required to send this frame (carrier sensing).
  - Duration: channel time reserved for acknowledgements or next frames (virtual carrier sensing).

B. Trace Preparation

1) Channel error rate: Because at any point in time t, we can only scan a single rate r with RTS/CTS on or off, if a rate adaptation algorithm chooses to send a packet at time t using any rate other than r and/or using a different RTS/CTS configuration, we will not have a corresponding packet in the collected trace. Therefore, we need to determine or compute information about all packets that could have been sent starting at time t. We call this phase trace preparation, because we use information from the collected trace (which contains only information about rates that were used and packets that were sent) to prepare a complete trace (which contains information about all rates that could have been used).

To predict if a missing packet could be received or not, our prototype uses a window of 100 milliseconds (50 before and after) to compute packet reception probability. This is computed based on the number of packets sent and received within that time window. At time t, for each rate r, and for RTS/CTS on and off we consider only packets from the collected trace that are within the window, were sent at rate r and which were sent with RTS/CTS on and off, respectively. The computed probability is then used to predict if a missing packet would be received or not. That is, for each rate r we record whether or not the packet would have been received with RTS/CTS on and off. This information is recorded in the complete trace.

2) Channel access: The delay caused by non-WiFi interferers is stored in the ETT, while third-party WiFi traffic received by the sender is stored in the TWTT. The TWTT is used directly by the trace processing engine. On the other hand, the trace processing engine needs to know the delay caused by non-WiFi interference for a packet ready to be transmitted at time t. Unfortunately, we may not have such a sample in the ETT. We use the $100\ ms$ time window to compute the expected delay at time t, using a technique similar to that used for estimating the fate of a missing packet. Our evaluations show that this technique accurately predicts the expected delay caused by non-WiFi interferers.

3) Additional data: Signal Strength: We currently estimate the signal strength of a missing packet using a window of two packets. We use a linear interpolation of the RSSI (Received Signal Strength Indication) of the two surrounding packets. The Atheros chipset used in our prototype has a noise floor of -95 dBm, so the SNR can be determined from the RSSI. To evaluate the accuracy of our RSSI estimation technique, we adapt the methodology used in [9]. We down sample the RSSI data (i.e., remove data) obtained from an actual experiment to create a trace which is similar the collected trace (i.e., it is missing the same amount of information as a collected trace). Then, we apply our RSSI estimation algorithm to the down sampled trace to prepare a complete trace. We then compare the information from the complete trace with that from the actual experiment. In an experiment conducted using a mobile device moving at the speed of someone walking, we observed that over 85% of the estimated RSSI values are within 1 dBm of the actual values and that the error is generally bounded by a few dBm. Although the trace used for this experiment was obtained from a challenging mobile experiment with highly variable channel conditions, our heuristic is highly accurate.

Clearly, other approaches and different parameters could be used during trace collection and preparation. Although the evaluation of our prototype in Section V demonstrates that our prototype produces highly accurate results for the scenarios we have tested, in future work we plan to explore some of these choices. For example, the choice of window length is a design decision that depends on the variability of the wireless channel conditions and different window sizes may be needed in environments with faster moving mobile devices or more transmission rates (e.g., in 802.11n).

C. Trace Processing

During the trace processing phase, a rate adaptation algorithm is simulated using the complete trace. The complete trace

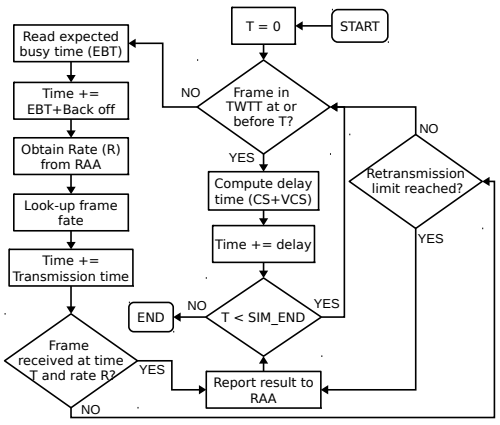


Fig. 4: Simulation flowchart

consists of the prepared experiment traffic trace (P-ETT) and the third-party WiFi traffic trace (TWTT). Trace processing involves tracking the current (simulated) time and determining which rate would be chosen by the given algorithm. Figure 4 illustrates the approach. Before the transmission of any frame, the simulator first checks the third party traffic trace (TWTT) to simulate the delay cause by receiving WiFi frames at the sender. If any frame was observed at time T, the delay caused by receiving this frame is calculated. This includes the time the received frame is in the air (carrier sensing) plus any additional time for which the frame reserves the channel (virtual carrier sensing). Then, the simulation time (T) is updated and the TWTT trace is checked again for any unprocessed third party frames at or before time T. We consider the unprocessed frames in the TWTT before time T, because third party transmissions that started before T also affect channel sensing. When no such frames exist, the simulator proceeds to the transmission of the next frame.

The next step is to simulate the delay caused by non-WiFi devices while performing carrier sensing. This delay is directly read from the complete trace. Additional delays caused by the DCF protocol, such as back-off time, are computed and the simulation time is updated. The trace processing engine implements DCF inter frame spacing to achieve realistic timing and to ensure the accurate computation of throughput. The simulator looks in the complete trace to determine the fate of the frame sent at time T, rate R (specified by the RAA), and with the RTS/CTS option on or off. To accurately compute packet transmission times, we utilize information about the structure of OFDM frame formats and the transmission time of each segment (i.e., header and payload). If the frame transmission fails, the frame is retransmitted until the frame is successfully transmitted or the retransmission limit is reached. The result is logged and is reported to the RAA. The logged data includes the rate used and whether or not the frame was received. We then post-process the log file to better understand an algorithm's behavior and to compute statistics related to performance such as the average throughput over different time scales and the total number of bytes received.

D. Limitations

During the trace collection phase, we sample all rates as quickly as possible to maximize the number of collected samples. Although we are able to simulate delays before sending packets due to third party traffic, when contention is too high the sender's access to the channel will be limited and may not be able to obtain enough samples to produce realistic complete traces. In some of our experiments the sender's ability to access the channel is limited and fewer samples are obtained. Although we are able to obtain highly accurate results in several interesting scenarios in this paper (see Sections V-A and V-C), further work is needed to understand the number of samples required to obtain realistic results.

IV. Evaluation Methodology

The purpose of our evaluation section is not to evaluate different RAAs but instead to evaluate the prototype implementation of T-RATE and to demonstrate that it can be used with a variety of RAAs. When possible, we compare the throughput obtained by T-RATE with throughput from an actual experiment. This requires us to be able to construct scenarios where we can ensure that the channel conditions are repeatable. Thus, we conduct some experiments in a controlled environment where there is no interference from devices other than those we use to generate interference.

For some evaluations, we compare the throughput obtained with an actual experiment run using a fixed MAC data rate with that obtained by the trace-processing engine using that same rate. The reasons for this approach are: (1) At any point in time a rate adaptation algorithm may choose any one of the available data rates. This ensures that for any rate chosen by any algorithm the trace-processing engine will match the throughput obtained in the experiment. (2) It is easier to reason about the expected throughput for a fixed rate than for an algorithm that is constantly adjusting the rate. The rates available in 802.11g are 6, 9, 12, 18, 24, 36, 48, and 54 Mbps. Most of our graphs show only rates of 6, 24 and 54 Mbps, because other rates show similar behaviour.

Despite using carefully controlled and monitored environments, we could not obtain repeatable results for some experiments (e.g., for some hidden terminal and mobile experiments). Additionally, we are interested in evaluating our prototype in environments where repeatability is not possible. For such scenarios, we use the following approach. Because the sending device transmits frames as fast as possible during the trace-collection phase, an experiment is being conducted during trace collection. The rate adaptation algorithm that is used repeatedly cycles (pseudo-randomly) through each rate. Therefore, we implement a similar algorithm in the trace-processing engine. The difference is that in the trace-processing engine we ensure that the pseudo-random selection of rates happens in a different order from that in which the trace is collected. This ensures that the trace-processing engine must use information from frames added to the complete trace (during the trace-preparation phase) that are not present in the collected trace.

A. Equipment and Software Used

We continuously monitor all of our experiments using an AirMagnet Spectrum XT [2]. This analyzer is able to find and classify WiFi and non-WiFi interference. Our 802.11g networks are constructed using netbooks and laptops running Linux with the Atheros Ath9k driver. We use a 2.4 GHz FHSS

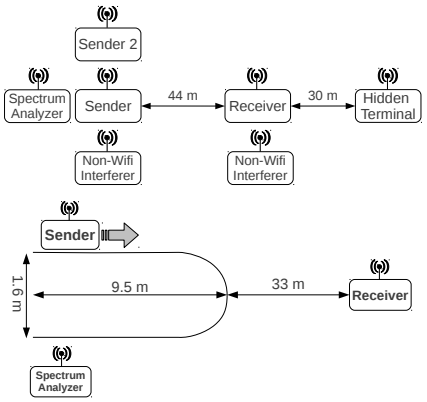


Fig. 5: Stationary (top) and mobile (bottom) experiment setups

device (a controller for an RC helicopter) to generate non-WiFi interference. This device was chosen because it has a USB interface which allows us to programatically turn the device on and off at precisely the same time when repeating experiments. We use Iperf [8] to send UDP packets as fast as possible between the sender and the receiver. When plotting throughput versus time, each point shows the average and 95% confidence intervals of 10 half second measurements.

B. Different Evaluation Scenarios

Figure 5 illustrates the different devices and their relative positions in our stationary and mobile scenarios. Not all devices are used in all experiments and the details of each experiment are provided in the evaluation section. In mobile scenarios, we use an electric train to carry a notebook computer (the mobile device). A train is used to ensure that the same path is traveled at the same speed (i.e., approximately walking speed) for multiple runs of the same experiment. The train moves on a half-oval shaped track. Every time the train reaches one end one of the track the direction is reversed.

V. Evaluation

To conduct a comprehensive evaluation of our prototype, we consider the *environmental factors* that can affect the throughput of an 802.11 network as shown Figure 1. In Section V-A, we evaluate our prototype's ability to obtained correct throughput when the sender's channel access is affected. In Sections V-B1 and V-B2, we evaluate the accuracy of T-RATE using scenarios where the channel error rate is affected due to interference.

A. Channel Access

We now evaluate the realism and accuracy of T-RATE when channel access is limited because of WiFi and non-WiFi interference near the sender.

1) WiFi Source: **Experiment Setup:** In addition to the Sender sending to the Receiver (Figure 5 – top), Sender 2 also sends UDP traffic to the same Receiver at a steady rate of 3 Mbps, roughly mimicking a video stream. This traffic is injected from time 40 to 70 seconds. Results of these experiments are shown in Figure 6(a).

This scenario mainly targets the processing engine's ability to implement carrier sensing and virtual carrier sensing caused by third party traffic. Figure 6(a) is obtained by conducting four experiments. The environment is controlled and monitored to ensure that channel conditions are repeatable across each experiment. One experiment is run to collect a trace and the others are to run each of the three fixed rates. The throughput obtained using each of the three fixed rates is compared with that reported by the trace-processing engine when using those same fixed rates.

The tight match between the trace-driven and experimental results suggests that the processing engine is able to handle this scenario with high precision. As expected, the throughput of the rates 6 and 24 are only slightly affected while WiFi interference is present (from 40 to 70 seconds) while the throughput of the 54 Mbps experiment has a more noticeable 3 Mbps drop. Also note that when the environment is *interference-free* (prior to 40 and after 70 seconds), the throughput obtained for each rate matches the theoretical throughput expected.

2) Non-WiFi Source: Non-WiFi transmitters can also cause the carrier sensing protocol to report busy which can decrease throughput by introducing delays before a packet can bet sent.

Experiment Setup: A non-WiFi transmitter (Figure 5 – top) near the Sender operates from time 40 to 70 seconds. Results from this experiment are shown in Figure 6(c).

Figure 6(b) and (c) show the importance of the channel cycle information (CCI). When CCI is not used (Figure 6(b)) the throughput obtained using T-RATE does not closely match the experiment. However, Figure 6(c) shows the excellent match obtained for all rates by including CCI in T-RATE.

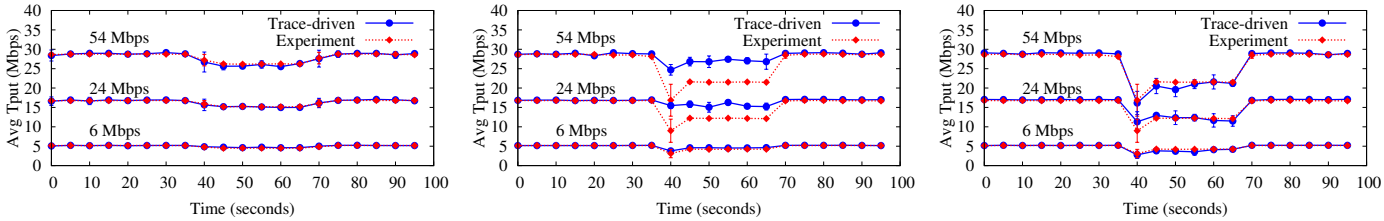
B. Channel Error Rate

A sender's throughput also depends on the channel error rate. A channel's error rate can be influenced by two important factors: interference (which can come from WiFi and non-WiFi sources near the receiver) and signal propagation (i.e., path loss and fading). In this section, we examine the accuracy of T-RATE using scenarios where these influences are present.

1) Non-WiFi Interference: **Experiment Setup:** In this experiment we place a non-WiFi interferer near the Receiver (Figure 5 – top) and out of range of the Sender, so that it does not affect channel access. The non-WiFi transmitter operates from time 40 to 70 and the results are shown in Figure 7(a).

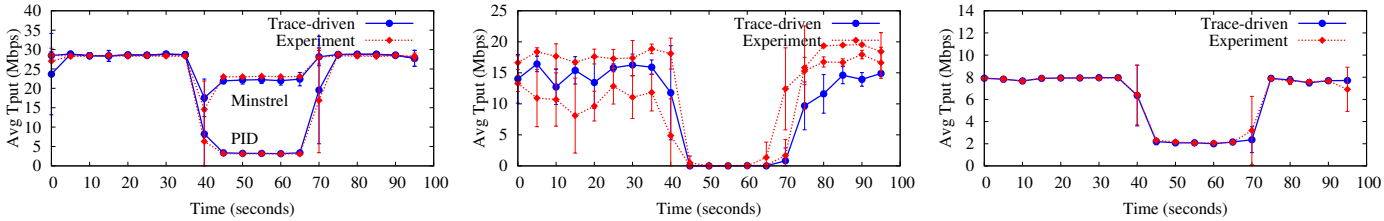
Figure 7(a) compares the throughput obtained from real experiments using the PID (Proportional-Integral-Derivative) and Minstrel algorithms from the Linux Ath9K driver with throughput obtained using the trace-driven evaluation of these algorithms. These graphs show that for both algorithms, there is a tight match between the experiments and those obtained with T-RATE. The RAAs implemented in the trace-processing engine change rates at times that closely match the experiment. This suggests that the trace-processing engine correctly implements these algorithms and that the information required by and available to them in the complete trace is also accurate.

2) WiFi Interference: **Experiment Setup:** The Hidden Terminal WiFi transmitter in Figure 5 (top) transmits a 3 Mbps UDP stream for 30 seconds starting at time 40. The Hidden Terminal is configured to use a 6 Mbps MAC rate in



(a) (Virtual) Carrier Sensing (b) Carrier Sensing, without CCI (c) Carrier Sensing

Fig. 6: Channel Access



(a) Non-WiFi, PID, RTS off (b) WiFi, PID, RTS on (c) WiFi, Round-robin, RTS on/off

Fig. 7: Channel error rate due to interference

this scenario in order to maintain connectivity with the access point. Results from this experiment are shown in Figure 7(c).

In the area of the building we used to conduct our experiments, we needed to lower the transmission power of the Sender in order to create a hidden terminal scenario. As a result, the throughput obtained when using fixed rates of 48 and 54 Mbps were consistently zero due to a low SNR. Unfortunately, for the same reason, the throughput obtained with 24 and 36 Mbps is very unstable and was not repeatable across experiments. Since RAAs may choose rates of 24 or 36 Mbps the throughput is also very unstable when a real RAA is in place. Figure 7(b) shows that for two identical runs of an experiment using PID, the throughput can be significantly different. Thus, under this scenario, it is not possible to compare results obtained from experiments with those obtained from the trace-driven framework.

Because channel conditions change significantly from one experiment to the next, the experiments can not be repeated. Therefore, we can not expect the results obtained using the trace-driven framework to match those from experiments. Instead, we utilize the alternative method for evaluating our results described in detail in Section IV. That is, we compare the results obtained during the trace collection experiment with those obtained using the trace-driven framework while implementing a similar RAA. The excellent match between the experimental and trace-based throughput shown in Figure 7(c), suggests that T-RATE handles interference caused by a hidden terminal accurately.

3) Stationary Path Loss: **Experiment Setup:** We periodically change the transmission power of the Sender in a controlled fashion (Figure 5 – top). The transmission power starts at 17 dBm and is reduced by 1 dBm every 3 seconds until it reaches 0 dBm. It is then increased by 1 dBm every 3 seconds until 17 dBm is reached. Figure 8 shows results from experiments conducted using this scenario.

This experiment is designed to evaluate T-RATE's ability to handle frame loss due to path-loss and to evaluate the trace-processing engine's implementation of RBAR [7]. RBAR proposes modifications to the 802.11 standard to transmit SNR information back to the sender. The sender then uses a table lookup to determine an appropriate rate for a given SNR. RBAR is implemented to demonstrate that T-RATE can accommodate SNR-based algorithms. To test the correctness of the operation of RBAR in our trace-processing engine, we have provided RBAR with the values shown in Table III. We constructed this table to cover the RSSI ranges experienced in this experiment and to force RBAR to select all rates.

RSSI	Rate	RSSI	Rate	RSSI	Rate	RSSI	Rate
≥ -58	54	≥ -60	48	≥ -62	36	≥ -64	24
≥ -66	18	≥ -68	12	≥ -70	9	≤ -71	6

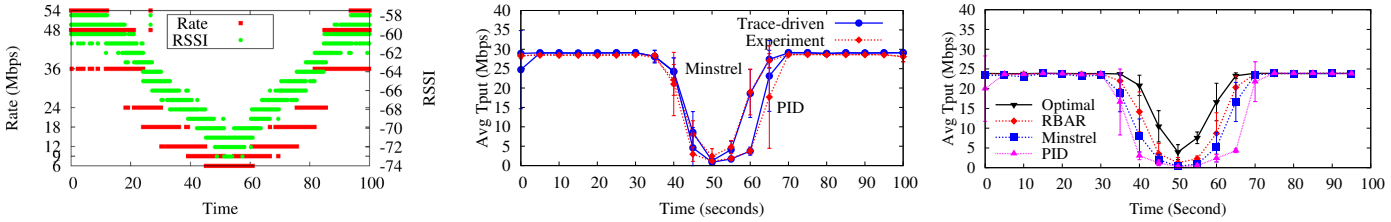
TABLE III: RBAR rate selection table

Figure 8(a) shows how the RSSI of the received frames (right y-axis) changes as the result of changing the transmission power over time (the x-axis). The left y-axis shows the rates chosen by RBAR. This graph shows that the trace-processing engine's implementation of RBAR operates correctly based on Table III.

Figure 8(b) shows results obtained using the PID and Minstrel algorithms both experimentally and using T-RATE. The good agreement between the results obtained from the processing engine and experiments shows that T-RATE handles path-loss accurately for this scenario.

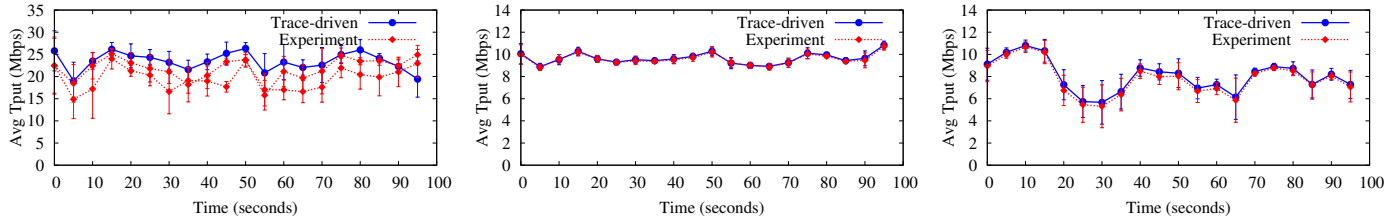
Figure 8(c) shows the throughput reported by T-RATE for the same experiment when RTS is on. As expected throughput is lower than in Figure 8(b) where RTS is off. We compare the throughput achieved by the PID, Minstrel, and RBAR algorithms with an algorithm that chooses the best possible rate at each moment in time (Optimal). RBAR, which has now been properly calibrated for this environment, outperforms the loss-based RAAs. This is as expected because it can react to changes in signal strength faster. The ability to determine the optimal rate at any moment in time is an additional strength of T-RATE and should provide insights in some scenarios.

4) Mobile Path Loss and fading: **Experiment Setup:** The scenario used is as shown in Figure 5. To repeat experiments a mobile device (the Sender) is placed on an electric train which



(a) RBAR's reaction to changes in RSSI (b) PID and Minstrel, RTS off (c) PID, Minstrel, RBAR, Optimal: RTS on

Fig. 8: Path loss (Tx power changing, stationary)



(a) Controlled Mobile, Minstrel (b) Controlled Mobile, RR, RTS on/off (c) Field trial, RR, RTS on/off

Fig. 9: Mobile

moves at approximately walking speed and results are shown in Figure 9(b).

Unfortunately, despite controlling for all environmental factors, including the exact path traveled, these experiments are not repeatable. Figure 9(a) shows the variability across two experiments and results obtained using our framework when the Minstrel algorithm is used. Because of the inability to repeat experiments we again rely on our alternative approach to evaluate T-RATE. These results are presented in Figure 9(b). The excellent match suggests that the results from the trace-driven framework are highly accurate despite the continual changes in signal strength that occur in this mobile scenario.

C. Field Trial

We now test our framework in an environment that is not under our control and is subject to many sources of interference, path loss and fading.

Experiment Setup: The receiver is placed in an office environment with multiple cubicles. The sending laptop is carried to different areas of the same office at walking speed. A spectrum analyzer is used to monitor natural interference (i.e., we do not inject any controlled interference) from WiFi and non-WiFi sources. Other settings are kept the same as in our controlled experiments and result are shown in Figure 9(c).

Figure 9(c) shows the throughput reported by the trace-driven framework and that obtained from an actual experiments where the transmission rate is changed in a round-robin fashion (recall that the trace-processing engine uses a different ordering from that used in trace-collection). The figure shows that the trace-driven results match those from the experiment.

We now further analyze the field trials results. The goal is to determine if T-RATE can produce accurate and realistic results when using algorithms other than round robin. Recall that if, for each available rate, at every point in time, the trace-processing engine produces the correct throughput, then no matter which rate is selected by an RAA it will obtain the correct throughput. For this reason, we examine results obtained via the trace-processing engine, PID and Optimal. The top graph in Figure 10 shows the throughput reported by T-RATE using different algorithms. To make the graphs more legible, we have excluded some constant rates, all confidence intervals and Minstrel.

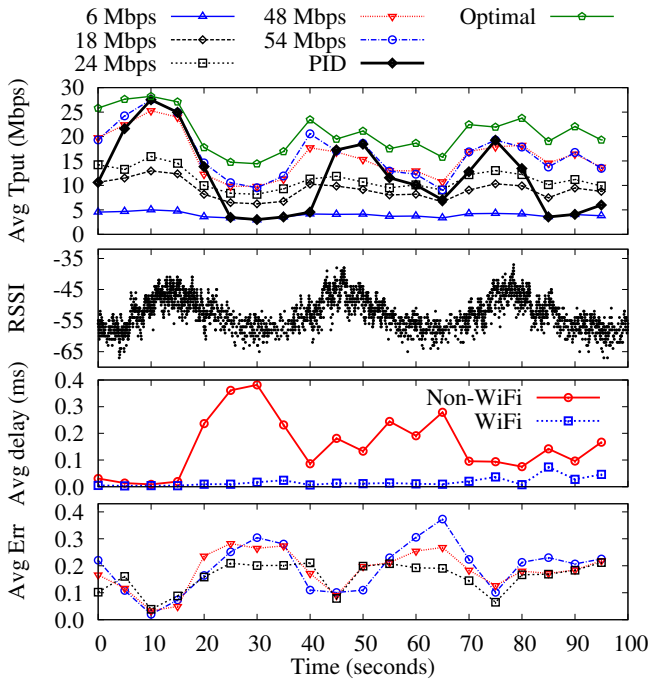


Fig. 10: Throughput, RSSI, Interference and Error Rate

Despite the significant variation in the throughput over time, the 54 Mbps rate provides the best throughput of all the rates (excluding the optimal choice). Except for at 65 seconds, where 48 Mbps provides the best throughput. To understand the variation in throughput, and why 54 Mbps provides the best throughput, we first consider the RSSI of received frames. The RSSI graph (second from the top in Figure 10) shows that the RSSI of the received frames are mostly greater than -60 dBm, so it is unlikely that any transmission rate will experience frame loss due to path loss. Therefore, any decrease

in throughput is due to other factors such as limited channel access and/or frame loss due to interference. Indeed, the spectrum analyzer reports that a microwave oven was active at times during this experiment.

The bottom graph in Figure 10 shows the average error rate for fixed rates of 24, 48, and 54 Mbps. Comparing the error rate graph with the throughput graph, we see that for each rate, as the error rate increases, the throughput decreases. At time 10 the error rate of all fixed rates is close to 0 and the throughput of all rates is maximized. The cause of the changes in the error rate can be seen in the graph that is second from the bottom. This "interference graph" shows the average expected delay per packet sent, caused by non-WiFi and WiFi interference. Although there is WiFi traffic, the average delay imposed by this traffic is low relative to that from non-WiFi sources. The shape of the average error rate graphs roughly corresponds to that of the non-WiFi interference graph.

The behavior of PID can be explained by examining the error rates. When the error rate exceeds a threshold, PID reduces the selected data rate. Because the cause of frame errors in this experiment is non-WiFi interference, reducing the transmission rate may not help. Figure 10, shows that as the error rate increases from 10 to 30 seconds, PID reduces its MAC data rate until it reaches 6 Mbps at time 30. It remains near 6 Mbps until after 40 seconds when the error rate starts to decrease and PID starts to choose higher rates.

Overall, the results obtained using our prototype and the behaviour of the algorithms are as would be expected given the environment and channel conditions under which the trace was collected. We believe that T-RATE is a significant step towards improving the evaluation of 802.11 RAAs.

VI. Discussion

One of the strengths of our approach is is that it captures data related to properties of the channel under conditions in which traces are collected. However, a trace is specific to the devices used when capturing the traces. We note that this is also an issue for existing trace-driven and experimental approaches for evaluating RAAs. We plan to study the degree to which results might be used across different devices and scenarios in future work. We also plan to: examine scenarios where mobile devices move at higher speeds, better understand and outline limitations that might be caused by limited channel access (e.g., due to dense WiFi use), to collect traces using a wide variety of scenarios, and to examine 802.11n and 802.11ac networks. We expect that T-RATE will translate relatively easily to 802.11n networks. However, new trace collection techniques may be required for devices with 3 or more antennas because of the large number of available rates.

VII. Conclusions

In this paper, we present the design, prototype implementation and evaluation of T-RATE, a trace-driven framework for evaluating 802.11 RAAs. We devise mechanisms that allow us to capture traces on communicating 802.11 devices, while conducting experiments under realistic conditions. We show that T-RATE can be used to conduct highly accurate evaluations of RAAs under a variety of channel conditions that are more representative of scenarios under which devices are likely to be used than previously possible. Moreover, our portable traces and trace-processing engine seamlessly couple with different algorithms to provide for easy, portable, repeatable, and realistic evaluations.

VIII. Acknowledgments

Brecht is partially supported by a Discovery Grant and an Accelerator Supplement from the Natural Sciences and Engineering Research Council (NSERC) of Canada. Ali Abedi was partially supported by a scholarship from Blackberry. Both authors would like to thank Tyler Szepesi, Doug Terry, S. Keshav for feedback on earlier drafts of this work.

References

[1] Acharya, P., Sharma, A., Belding, E., Almeroth, K., and Papagiannaki, K. Congestion-aware rate adaptation in wireless networks: A measurement-driven approach. In *SECON* (2008).

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

[3] Aniket, Carlsson, N., Williamson, C., and Arlitt, M. Ambient interference effects in WiFi networks. In *NETWORKING* (2010).

[4] Deek, L., Garcia-Villegas, E., Belding, E., Lee, S.-J., and Almeroth, K. Joint rate and channel width adaptation for 802.11 MIMO wireless networks. In *SECON* (2013).

[5] Giustiniano, D., and Mangold, S. Caesar: Carrier sense-based ranging in off-the-shelf 802.11 wireless lan. In *CoNEXT* (2011).

[6] Halperin, D., Hu, W., Sheth, A., and Wetherall, D. Predictable 802.11 packet delivery from wireless channel measurements. In *SIGCOMM* (2010).

[7] Holland, G., Vaidya, N., and Bahl, P. A rate-adaptive mac protocol for multi-hop wireless networks. In *MobiCom* (2001).

[8] IPerf. http://sourceforge.net/projects/iperf/.

[9] Judd, G., and Steenkiste, P. A simple mechanism for capturing and replaying wireless channels. In *ACM SIGCOMM workshop on Experimental approaches to wireless network design and analysis* (2005).

[10] Judd, G., Wang, X., and Steenkiste, P. Efficient channel-aware rate adaptation in dynamic environments. In *MobiSys* (2008).

[11] Lee, K., Navarro, J., Chong, T., Lee, U., and Gerla, M. Trace-based evaluation of rate adaptation schemes in vehicular environments. In *Proceeding of VTC '10* (2010), pp. 1 –5.

[12] Noble, B. D., Satyanarayanan, M., Nguyen, G. T., and Katz, R. H. Trace-based mobile network emulation. In *SIGCOMM* (1997).

[13] Ravindranath, L., Newport, C., Balakrishnan, H., and Madden, S. Improving wireless network performance using sensor hints. In *USENIX* (2011).

[14] Rayanchu, S., Patro, A., and Banerjee, S. Airshark: detecting non-WiFi RF devices using commodity WiFi hardware. In *IMC* (2011).

[15] Sadeghi, B., Kanodia, V., Sabharwal, A., and Knightly, E. Opportunistic media access for multirate Ad Hoc networks. In *MobiCom* (2002).

[16] Sen, S., Santhapuri, N., Choudhury, R. R., and Nelakuditi, S. AccuRate: Constellation based rate estimation in wireless networks. In *NSDI* (2010).

[17] Shen, W.-L., Tung, Y.-C., Lee, K.-C., Lin, K. C.-J., Gollakota, S., Katabi, D., and Chen, M.-S. Rate adaptation for 802.11 multiuser MIMO networks. In *Mobicom* (2012).

[18] TCPDUMP. http://www.tcpdump.org/.

[19] Tie, X., Seetharam, A., Venkataramani, A., Ganesan, D., and Goeckel, D. L. Anticipatory wireless bitrate control for blocks. In *CoNEXT* (2011).

[20] Vutukuru, M., Balakrishnan, H., and Jamieson, K. Cross-layer wireless bit rate adaptation. In *SIGCOMM* (2009).

[21] Wong, S. H. Y., Yang, H., Lu, S., and Bharghavan, V. Robust rate adaptation for 802.11 wireless networks. In *MobiCom* (2006).

[22] Zhang, J., Tan, K., Zhao, J., Wu, H., and Zhang, Y. A Practical SNR-Guided Rate Adaptation. In *INFOCOM* (2008).