The Dissertation Committee for Muhammad Owais Khan
certifies that this is the approved version of the following dissertation:

Improving the Performance and Efficiency of Wireless Networks using Rate Adaptation

Committee:

Sriram Vishwanath, Supervisor

Lili Qiu, Co-Supervisor

Gustavo de Veciana

Christine Julien

Mohamed Gouda

Improving the Performance and Efficiency of Wireless Networks using Rate Adaptation

by

Muhammad Owais Khan, BEE; MSE

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2015

Improving the Performance and Efficiency of Wireless Networks using Rate Adaptation

Publication No. ____________________

Muhammad Owais Khan, Ph.D.
The University of Texas at Austin, 2015

Supervisors: Sriram Vishwanath
Lili Qiu

Recent years have seen a staggering increase in the deployment and utilization of wireless networks. More and more devices are being equipped with Wireless LAN (WLAN) cards to take advantage of the omnipresence of WLAN networks. Therefore, it has become necessary that the protocols used by WLANs are efficient and provide good performance. Rate Adaptation protocols are an important mechanism employed by WLANs to improve network performance. This dissertation develops three complementary techniques, which use rate adaptation to optimize and improve performance by i) performing rate adaptation to optimize energy consumption, ii) developing a more accurate technique to predict the frame delivery ratio that is used by rate adaptation protocols, and iii) jointly optimizing rate adaptation with data retransmission to maximize throughput.

More specifically, in i), we use extensive measurements to develop a simple yet accurate energy consumption model for 802.11n wireless cards. We use the model to drive the design of an energy aware rate adaptation scheme. A major benefit of a model-based rate adaptation is that applying a model allows us to eliminate frequent probes required in many existing rate adaptation schemes.

In ii), we find that the accuracy of existing delivery ratio calculation techniques is still limited due to bursty errors inherent to the wireless channel. We develop a new method for computing packet delivery rate that captures the burstiness of errors. Furthermore, we propose a new data interleaving technique, which leverages our framework to reduce the burstiness of errors, thereby improving frame delivery ratio.

Finally, in iii), we address the susceptibility of wireless networks to transmission failures due to dynamic channel conditions and unpredictable interference. To efficiently recover from failures, we propose a retransmission scheme where the receiver combines information received from multiple failed transmissions associated with the same frame. The protocol has two distinguishing features. First, it simultaneously supports partial retransmission and combines bits with low confidence. Second, it jointly optimizes the data rate of the retransmission and the information to be retransmitted to maximize throughput.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The last decade has seen an explosive growth in the deployment and use of Wireless Local Area Networks (WLANs). Owing to its low cost and ease of deployment it has become the de-facto method to providing internet access in a large array of setting; from providing wireless connectivity at home to providing coverage across university campuses. In recent years, WLAN capability has been added to consumer electronic devices. This trend is likely to continue with Internet of Things (IOTs) on the horizon [48], resulting in billions of new devices getting connected to WLANs. Recent studies indicate that there will be more than 30 billion wireless capable devices by 2020 [21]. Most new technologies that incorporate wireless support are small devices which run on battery. As a result power consumption is a key factor in these devices.

Energy Consumption:

A recent study shows [76] that radio interfaces in smartphones can account for 50% of the power budget under typical use. In [37], Halperin *et al.*show that IEEE 802.11n wireless cards can draw up to 2.1W in MIMO configurations. Furthermore, using large channels worsens the power consumption by 40%. With the continuing increase in the number more antennas and wider channels, the power consumption of these devices is likely to go up.

Throughput Performance:

The growth of wireless devices has resulted in an exponential growth of traffic demand. Mobile data traffic in 2013 was nearly 30 times the entire global internet traffic in 2000 [18]. The growth is also not showing any signs of slowing down. There was an 69% increase in mobile data traffic in 2014 compared to 2013. 46% of the traffic generated by mobile devices was carried by WLANs. It is projected that 72% of the world's mobile data traffic will video with WLANs carrying 50% of the traffic. Therefore, it is imperative to that new WLAN devices have the capability to support high data rates and provide the best throughput performance.

Thesis Contributions: As the number of wireless devices continue to increase, it is necessary that new protocols are developed to keep up with the growing trend of wireless traffic. In this thesis, we approach these problems from the context of rate adaptation and propose three new schemes which i) reduce the energy consumption, ii) provide better delivery ratio estimates for more accurate rate adaptation and iii) improve throughput by leveraging retransmission.

Before, delving into details in the subsequent chapters, here we provide a brief overview of each of these approaches.

1.2 Minimizing Energy Consumption

Overview: Rate adaptation has been an active area of research for the last decade. While most work has focused on selecting the rate to maximize throughput, how to select data rate to minimize energy consumption is an important yet under-explored topic. This problem is becoming increasingly important with the rapidly increasing popularity of MIMO deployment, because MIMO offers diverse rate choices (*e.g.*, the number of antennas, the number of streams, modulation, and FEC coding) and selecting the appropriate rate has significant impact on power consumption.

While MIMO provides a large capacity gain, using multiple antennas can consume significantly more energy, which is undesirable for mobile devices [37]. For a fixed number of antennas, reducing the transmission time always results in a decrease in energy consumption. But for the same transmission time, the energy consumed by multiple antennas is much higher than a single antenna. This is because MIMO transmission requires additional hardware and RF chains for MIMO processing, which increases energy consumption. On the other hand, using multiple antennas reduces transmission time by allowing multiple data streams to transmit simultaneously. Hence, there is a trade-off between minimizing the transmission time using multiple antennas and the additional energy cost associated with using multiple antennas.

Figure 1.1 compares transmission time of a single antenna with that of using two and three antennas. The plot is based on the transmitter energy model for Intel 5300 WiFi card, which is presented in Section 3.1.2. The x-axis

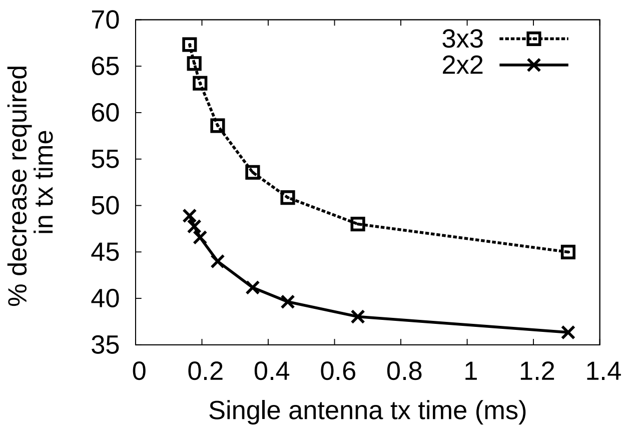


Figure 1.1: % reduction in transmission time for MIMO needed over SISO for energy improvement.

shows transmission time of a single antenna transmission. The y-axis shows the percentage of transmission time that two and three antenna MIMO transmissions must reduce in order for them to have the same energy as the single antenna transmission. From the figure, we can see that for a single antenna transmission time of $0.2ms$, using 3 antennas is only beneficial if the transmission time can be reduced by more than 68%. In comparison, for transmission time of $1.3ms$, the number reduces to 50%. So in the best case scenario where the three antenna MIMO transmission uses the same modulation and coding rate as the single antenna transmission but transmits three streams, the transmission time will decrease by 66% and exceed the minimum required 50% reduction in transmission time, therefore leading to energy saving.

Challenges: The above example indicates that there is no single setting that minimizes energy in all cases and a single antenna does not always lead to minimum energy. The exact rate and antenna configuration that minimize energy depends on a number of factors, such as channel condition, wireless card energy profile, and frame size. Therefore it is essential to have a comprehensive understanding about how energy consumption relates to these factors and design a rate adaptation scheme that automatically selects the rate to minimize energy according to the current network condition and wireless device.

Contributions: Our goal is develop a rate adaptation scheme which selects rate to minimize energy. This requires developing an energy prediction model which can be used by the rate adaptation scheme. In an effort to achieve this goal, we make the following contributions.

- *Energy Measurements:* We conduct extensive measurements using different wireless cards to understand the relationship between the data rate and resulting energy consumption. Our main observation is that for a fixed number of antennas, the energy consumed in transmitting or receiving a frame is proportional to the expected transmission time (ETT) [24] (*i.e.*, the total amount of time required to successfully deliver a frame to the receiver), and the slope of the energy consumption versus ETT depends on the number of antennas being used.

- *Energy Model:* Based on these insights, we develop a simple yet accurate model to predict the energy consumption under different transmission and reception configurations.

- *Rate Adaptation:* We develop a model-driven rate adaptation scheme on top of the model to select the rate that optimizes energy consumption. Furthermore, we extend the rate adaptation scheme to support a trade off between energy and throughput.

Finally, we use extensive trace-driven simulation and real testbed evaluation to show the energy savings of our schemes compared to the existing approaches.

1.3 Delivery Ratio Estimation

Overview: The signal-to-noise ratio (SNR) is the gold standard metric that captures the quality of a wireless link. However, it is well known that it lacks predictable power; the performance of a wireless link under the same SNR can be dramatically different. This significantly complicates a wide range of wireless network management tasks, such as rate adaptation, scheduling, routing, and diagnosis.

Recently, [38] shows that the fundamental reason that SNR fails to predict wireless performance is frequency diversity. Due to frequency selective fading and narrow-band interference, the signal-to-noise ratio across even a 20-MHz WiFi channel can vary significantly. The frequency diversity is only going to increase, as the channel width further increases in order to satisfy explosive growth of traffic demands. Based on this observation, [38] develops a new metric, called effective SNR, which offers better predictability over SNR.

Effective SNR is computed by first getting SNR across each OFDM subcarrier based on Channel state information (CSI), then mapping SNR to BER for each subcarrier, and finally finding the SNR that has the same BER as the average BER across the subcarriers. Owing to its significant performance benefit over SNR, effective SNR has become a widely adopted metric for wireless channel quality and used as the basis for many recent rate adaptation schemes (*e.g.*, [38, 34, 54, 82]).

While effective SNR improves over SNR, we find its accuracy in predicting wireless link performance is still inadequate. This is because effective SNR only captures average BER before FEC decoding, but for wireless decoders, such as the widely used Viterbi decoder, not only does average BER matter, but also the burstiness of the corruptions is important. For the same average BER, if the corrupted bits are spread far enough apart, the decoder may achieve 0 error. In comparison, the decoder will incur a higher error when the corrupted bits are closer together.

A natural question is why we care about burstiness given that interleavers are widely used in practical systems. However, we find that the standard interleaver used in 802.11 does not completely remove the burstiness of corrupted bits. Using traces collected from WiFi networks using Intel WiFi Link 5300 (iwl5300) IEEE a/b/g/n wireless network adapters, we show that errors remain bursty even after interleaving. This is because the existing interleaver spreads immediate neighbors to 4 subcarriers apart, which may still experience correlated corruptions.

Challenges: Based on these observations it is obvious that a new technique is needed which can estimate the delivery ratio in the presence of bursty errors. The key challenge is to design a technique which captures the burstiness at a fine granularity but is computationally not too expensive to run. Furthermore, it needs to be general and applicable to different Modulation and FEC coding rates.

Contributions: We propose two delivery ratio estimation techniques that capture the burstiness of errors. Furthermore, we apply our approach to designing a better interleaver. Our contributions can be summarized as follow:

- We use measurement to show that wireless errors remain bursty even after applying the standard WiFi interleaver. This may lead to inaccurate delivery rate estimation and sub-optimal performance in rate adaptation.
- We develop two complementary methods that compute frame delivery rates in the presence of frequency diversity and bursty errors. The first method gives insight into what leads to frame corruption, and the second method is more efficient.
- We design a new CSI-aware interleaver to reduce bursty error and improve decoding rate.
- We further develop a rate adaptation scheme based on our delivery rate estimation. It is flexible and can support different interleavers.

- Using extensive evaluation, we show that both our delivery rate estimation are accurate and significantly out-perform effective SNR. Moreover, our interleaver and rate adaptation built on top of our estimation out-perform existing interleaver and rate adaptation.

1.4 Smart Retransmission

Overview: Traditional rate adaptation schemes try to select the rate that maximizes throughput for the ongoing transmission. If the transmission fails, the rate adaptation treats the utility of the transmission to be 0 and discards the entire frame. This usually results in conservative rate selection so that a frame can be delivered in one attempt.

In reality, transmission failures occur due to a few bits in error and most bits in a failed transmission are received correctly. A few schemes like PPR [50] have been proposed, which use PHY-layer hints to identify the bits that are likely to be in error and only retransmit those bits. However, PPR does not provide a rate adaptation scheme. When traditional rate adaptation schemes are used with PPR, they make rate selection so that the transmission is successful in a single attempt thus negating an attempt any benefit from PPR.

We observe that multiple transmissions associated with the same frame should be considered as one unit and our goal is not to find the best for a single transmission but to maximize the throughput for the successful delivery of one frame. This means that one frame may be delivered over multiple transmissions. The transmission time depends on the data rate and the amount of data being transmitted. One can transmit the frame at a higher rate for the first transmission and then retransmit the corrupted bits in retransmission and have a smaller transmission time than the frame transmitted at a low rate in one transmission.

Challenges: Existing schemes (*e.g.*, PPR [50] and SOFT [98]) show significant performance benefit under a fixed data rate. However, their gain diminishes using the standard rate adaptation, which tends to pick a conservative rate and leaves little opportunity for partial retransmission and combining. The main challenge is to determine how partial retransmission and combining should be incorporated with rate adaptation.

Contributions: We aim to design a rate adaptation scheme which leverages partial retransmission to achieve higher throughput. In a effort to achieve this goal we make the following contributions.

- We study different combining options and show combining after demodulation but before FEC decoding achieves the best of both worlds – high combining gain and high flexibility (*i.e.*, supporting partial retransmissions and different data rates for retransmissions). We further design a practical approach to perform partial retransmission and combine bits based on the log-likelihood estimation.

- We develop a novel combining-aware rate adaptation scheme to effectively harness combining gains by taking into account the utility of a failed transmission and selecting the rate with the minimum total transmission and retransmission time of a frame. Our approach uses the standard data rates in IEEE 802.11 and is easy to deploy.

- We evaluate the performance of our scheme using trace-driven simulation. We demonstrate its effective under static and mobile scenarios and in the presence of interference.

1.5 Dissertation Outline

The outline of this proposal is as follows. In Chapter 2 we discuss the related work. Chapter 3 describes a rate adaptation protocol, which uses our energy model to minimize energy consumption. In Chapter 4, we propose two schemes to estimate the delivery ratio of a frame with bursty errors. Furthermore, we design a new interleaver to reduce bursty errors. In Chapter 5, we propose a rate adaptation protocol that utilizes partial retransmission and aggressive rate selection to maximize throughput. Finally, Chapter 6 concludes the dissertation.

Chapter 2

Related Work

In this chapter, we discuss the related work in more detail. We present the existing rate adaptation techniques for WLANs in section 2.1. In section 2.2, we focus on the work related to energy consumption in wireless networks. Finally in section 2.3, we discuss the existing schemes, which use combining and retransmissions to improve throughput performance of WLANs.

2.1 Rate Adaptation

Rate adaptation has received significant research attention in recent years (*e.g.*, [10, 13, 38, 43, 71, 86, 87, 94, 97]). There are a number of rate adaptation schemes that use loss rates for rate selection. For example, SampleRate [10] uses probes to select the rate that minimizes the expected transmission time. ONOE [71] in MadWiFi estimates long-term loss rate and uses thresholding for rate selection. RRAA [97] uses a short-term loss rate estimation for rate adaptation. MiRA [75] extends the loss rate based rate adaptation to MIMO by proposing a ZigZag search method to find the rate to optimize throughput.

Loss rates require a significant number of transmissions in order to measure accurately, so it cannot keep up with the changing wireless channel. Moreover, collisions and fading may both contribute to frame losses, and we should only reduce data rate upon fading, but not upon collisions [96]. So loss based rate adaptation require a scheme that can effectively distinguish the reasons for the losses in order to be effective.

SNR-based schemes are attractive because one can select rate based on SNR of a single frame instead of a large number of frames. The traditional SNR based schemes [43, 102] use average SNR to select the data rate, and are well known to yield inaccurate selection. More recently, [38] explains frequency diversity is the main reason that makes average SNR fail to accurately predict delivery rate. [38] proposes effective SNR, and shows it gives more accurate prediction of link performance and develops a rate adaptation based on it. Moreover, it only requires CSI, and CSI feedback is supported in the IEEE 802.11n standard. In addition, it not only supports SISO, but also supports MIMO by using post-processed CSI after MIMO processing. Turborate [88] extends the SNR based rate adaptation to multi-user MIMO scenarios.

There are also a number of approaches that leverage physical layer information. For example, SoftRate [94] develops a rate adaptation that uses PHY-layer hint. [87] leverages the dispersion between the transmitted and received symbols to derive the rate at which the packet could have been transmitted.

All the above works, however, focus on maximizing throughput and do not consider energy consumption. [56] is one of the few that considers energy in rate adaptation. It formulates the MIMO-OFDM minimum energy link adaptation problem as a geometric programming (GP) problem with an augmented parameter set under the control of the link adaptation protocol, but they do not empirically measure or derive energy models for wireless adapters. [60] also studies rate adaptation to reduce energy consumption. But unlike our work, which optimizes power based on the energy model, [60] uses probes to search for the rate that reduces energy.

Several rateless codes have also been proposed for wireless communication, such as LT [63], Raptor [89], Strider [34], and Spinal [77] codes. Some of these rateless codes have been shown to approach the Shannon capacity in real networks. However, the existing rateless codes have the following major limitations. (i) Rateless code has high complexity. For example, Strider's complexity is KN where N is the length of the packet and K is the block size and typically set to 33. Spinal code's complexity is a combination of $O((n/k)B * L * 2^{(k*d)})$ hashes and $O((n/k)B * 2^k)$ comparisons, where n is the total data, k is the block size, B is the beamwidth (*i.e.*, number of nodes kept at each step), d is the depth of the tree, n/k is the size each packet, and L is the number of packets transmitted before successful decoding. (ii) Rateless codes can incur large delay since they may require lots of transmissions to successfully deliver a frame and some of the rateless codes require block coding structure, which further increases delay, and (iii) Rateless codes are hard to deploy since they are significantly different from the modulation/demodulation schemes in the existing wireless systems.

2.2 Energy Measurement, Modeling and Optimization

A number of recent works have focused on measuring and analyzing the energy consumption of wireless cards and devices. Halperin *et al.* [37] study power consumption of the iwlwifi under different transmit power levels, card mode (*e.g.*, sleep, idle, transmit, receive), the number of active antennas and spatial streams, channel width and data rate. [2, 47] examine the power consumption of various WLAN cards and their impact on the battery life of the operating device. Feeney *et al.* [27] perform detailed measurement to obtain the energy consumption of an IEEE 802.11 wirless interface operating in an ad-hoc environment. Rice *et al.* [84] use a measurement framework to obtain fine-grained traces of a phone's power consumption, which are used to understand how different aspects of an application impact power consumption. Ebert *et al.* [78] measure the energy consumption of Aironet PC4800 wireless for different transmission rates, RF transmission powers and packet sizes. While the empirical observations works is insightful, they do not develop an energy model.

There has also been significant work recently that has focused on developing energy models based on emperical measurements. Carvalho *et al.* [15] present a simple model for power consumption in 802.11 ad-hoc networks as a function of the number of bytes and a constant radio overhead for all antenna configurations. They also augment it to account for channel contention costs. Balasubramanian *et al.* [4] present an empirical study of energy consumption on mobile phones for 3G, GSM, and WiFi energy consumption, and formulate an energy model for WiFi based on the transfer energy cost (per transfer size) and the maintenance cost of WiFi. Neither model considers the effects of multiple antennas, data rate, and transmit power. Sesame [23] is a system in which a mobile device creates its own energy model by using the battery interface with high accuracy. The scheme does not specifically model the energy consumption of the WiFi Adapter. Lochin *et al.* [62] use ACPI BIOS measurements two develop two models for the energy consumption of 802.11 networks. Carroll *et al.* [14] provide a detailed analysis of the power consumption of recent mobile phones. They also develop a power model of the Freerunner device that can be used for a number of usage patterns. Based, on these measurements they develop models to predict energy consumption under different scenarios. Rantala *et al.* [83], develop an energy model based on hardware measurements that allows the analysis and simulation of the energy efficiency of internet protocols ona wireless network interface. Garcia *et al.* [30] provide a detailed measurement study of energy consumption of wireless devices at a per-frame granularity. They show that a substantial fraction of energy is consumed across the protocol/implementation rather than during actual transmission. Furthermore, the authors use the measurements to develop an enery consumption model. The energy models developed in these papers differ from ours since they do not consider the impact of multiple antennas and MIMO transmissions. Furthermore, they also do not explicitly consider the impact of rate adaptation.

Some works use analytical models to analyze and optimize the energy consumption in WLANs. Ergen *et al.* [25] derive formulas to contrast the energy consumed in useful transmission and reception of data and the rest of the energy wasted due to overhearing. Jung *et al.* [52] propose an optimization of the power saving mechanism in which at the start of each beacon interval, the power saving mode periodically wakes for a dynamically selected duration. In *Miser* [81], Qiao *et al.*propose a scheme that minimizes the communication energy consumption in 802.11a/h by combining transmit power control and rate adaptation. Wang *et al.* [95] develop an anlytical model, which accounts for contending nodes, contention window, packet size and channel conditions to choose the optimum paramters to optimize the energy efficiency in WLANs.

Several works try to minimize time in idle listening mode. Rozner *et al.* [85] use virtualization techniques and energy-aware scheduling algorithm to reduce background traffic and allow 802.11 cards to enter Power Saving Mode (PSM) to save energy by 70%. Jang *et al.* [51] propose an energy management technique for 802.11n by configuring a client's sleep duration and antenna configuration.

Sleepwell [65] is a system that achieves energy efficiency by evading network contention among multiple APs in the vicinity of a mobile client. E-mili [103] is a scheme that reduces power consumed in idle listening by down-clocking radio. Catnap [22] allows a device to sleep by combining small gaps between packets into meaningful intervals, while [73] detects mobile phone bugs that prevent the phone from sleeping. DozyAp [40] allows power-efficient WiFi Tethering. Baiamonte *et al.* [3] identify the limitation of the current power saving mode (PSM) implemented in WLANs. They propose a distributed access scheme, whick allows a station to enter a low power operational state during channel contention by exploiting virtual carriers sensing and backoff functionality. Bruno *et al.* [12] define an p-persistent CSMA model based analytical framework to study the theoratical performance bounds of energy consumption. They propose a a feedback-based distributed algorithm to optimize the p parameter for optimal network performance. Chen *et al.* [17] present a contention adaptation mechanism to improve the energy efficienct of IEEE 802.11e EDCA by avoiding collisions and reducing the number of retransmissions. Garcia *et al.* [29] derive a closed-form expression for the optimal configuration of WLANs with respect to energy efficient fair channel access.

All these works are complimentary to our work, which focuses on optimizing MIMO transmissions to save energy.

2.3 Frame Combining and Partial Retransmissions

Several schemes have been proposed, which use spatial or temporal diversity to achieve better throughput. MRD [69] leverages receptions received at different APs to collectively recover the received frame. It considers the bits that are different in different receptions as corrupted and exhaustively searches over different combinations to find the one that passes CRC. SOFT [98] improves over MRD by using PHY-layer hints and combining different receptions using weighted averages instead of exhaustive search. [33] proposes a novel quantization scheme that allows different APs to efficiently share their received signals for combining. This is orthogonal to our focus. [50] uses PHY-layer hint to identify data with low confidence and only retransmit these data. It develops a dynamic programming approach to balance the feedback overhead and retransmission overhead. Unlike PPR, which combines the high-confidence bits from different transmissions, we observe that even low-confidence bits still contain useful information and further combine these bits using log-likelihood estimation. SOFT [98] also has an extension to exploit temporal diversity in the downlink by combining the original transmission and retransmission(s). However, it retransmits the entire frame instead of performing partial retransmissions.

There has also been significant theoretical work related to combining and partial retransmission. These techniques are usually grouped under Hybrid automatic repeat request (HARQ) systems. First ideas of incremental redundancy date back to Wozencraft and Horstein [99, 100]. The idea of combining was first proposed in 1977 by Sindhu [90]. Sindhu proposed to store the received packets, and then combine it with additional copies of the packet thus creating a more reliable single packet. Chase [16] extended the idea for code-combining systems. Lin and Yu [61] introduced Type-II HARQ, which alternates between message bits with Error Detection (ED) and error correction (FEC), thus creating the notion of incremental redundancy. Since then, there have been innumerable works that have proposed some variation on packet combining and incremental redundancy. Some of these works are [57, 70, 64, 91, 36, 58, 59]. A significant amount work has also been done in exploiting the diversity combining gains. Some of these schemes are represented by [67, 7, 68, 41].

While these works provide useful insights, they do not address the interaction between combining, partial retransmissions, and rate adaptation in WLANs.

Chapter 3

Energy Aware Rate Adaptation

3.1 Energy Measurements and Model

3.1.1 Measurement Methodology

To derive power models, we conduct fine-grained power measurements for the following wireless cards: (i) Intel 5300 N series wireless adapter [46], (ii) Atheros 802.11n wireless adapter, and (iii) embedded IEEE 802.11b/a WiFi device on a Windows Mobile smartphone with a single antenna. The first two are commonly used in laptops and can transmit or receive using up to three antennas. The third one is used to verify if the energy model carries over to the embedded WiFi device on a phone. Since multi-antenna WiFi devices for smartphones were not available in the market at the time of our study, we use a single antenna device.

To measure the power consumption of the wireless adapter cards, we use a desktop computer equipped with a PEX1-MINI PCI Express X1 Bus

[0]This chapter revises the following previously published material. Model-Driven Energy-Aware Rate Adaptation. Muhammad Owais Khan, Vacha Dave, Yi-Chao Chen, Oliver Jensen, Lili Qiu, Apurv Bhartia, Swati Rallapalli. ACM MobiHoc, Bangalore, India, August 2013. I was involved in development of the energy model and the implementation of the rate adaptation scheme. Vacha Dave and Oliver Jensen performed the energy measurements. Yi-Chao assisted in experimental evaluation. Swati Rallapalli and Apurv Bhartia collected channel traces for simulation evaluation. Lili Qiu, my advisor, supervised the entire project.

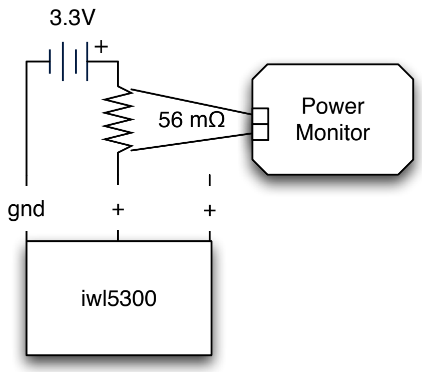
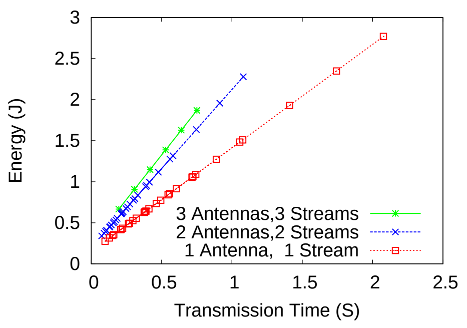


Figure 3.1: Circuit diagram of measurement setup for Intel card

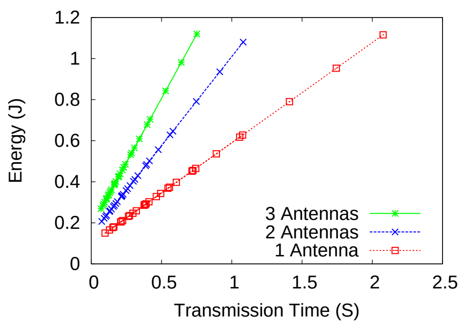
to PCI MINI Bus adapter [74]. It allows us to bypass the PCI bus power supply, and powers the wireless cards using an external source as shown in Figure 3.1. We supply the power to the wireless card using a Monsoon power monitor [79], which measures the current using a 56 milli-Ohm resistor. The power monitor samples instantaneous power at the rate of one reading per microsecond and returns a maximum power value for every 200μs period. We measure energy consumption of the embedded wireless adapter in a mobile phone by bypassing the battery and ground connector and supplying power to the phone as a whole using the same power monitor.

To control the frames involved in transmissions and to avoid unexpected frames, we use UDP packets, set retransmission threshold to zero, and turn off RTS/CTS. We vary data rate and antenna configuration by modifying device drivers of the Intel and Atheros cards. To force the phone into a particular data rate, we use HostAP daemon [45] as our access point and let it advertise only the required data rate in beacons.

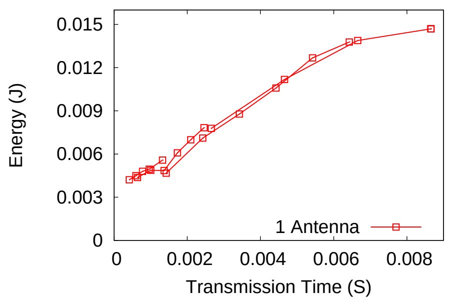
3.1.2 Measurement Based Model



(a) Intel transmitter



(b) Atheros transmitter



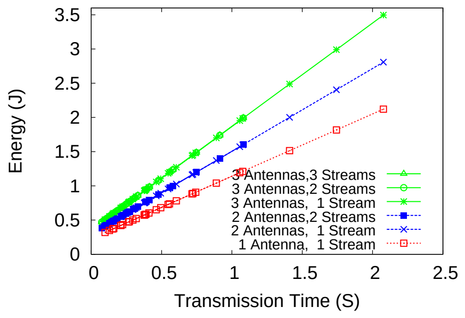
(c) Phone transmitter

Figure 3.2: Measured energy consumption under different transmission configurations as a function of ETT.

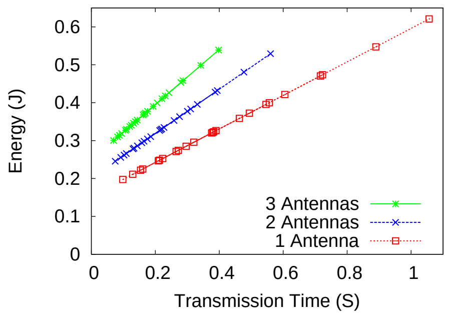
We collect and analyze power measurements from a variety of transmission and reception configurations. We vary the frame size from 250 to 1500 bytes. For Intel iwl5300 card, we collect power measurements for all high throughput (HT) 11n data rates using one, two, and three antennas supported by the card. The same process is repeated for the Atheros card and the phone. Figures 3.2 and 3.3 plot the energy consumption versus the expected transmission time (ETT) [24], which is defined as the expected time required to successfully transmit the frame from the source to the destination. *ETT* can be computed as

$$ETT = \frac{s}{r}\frac{1}{1-p},$$

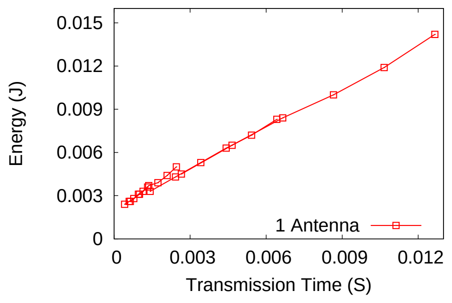
where p denotes the frame loss rate, r denotes the data rate, and s denotes the frame size. As we can see, in all the figures, the energy consumption is proportional to the expected transmission time (ETT) [24].



(a) Intel receiver



(b) Atheros receiver



(c) Phone receiver

Figure 3.3: Measured energy consumption under different reception configurations as a function of ETT.

The slope of the line depends on the number of transmitting and receiving antennas being used. This holds for all three cards we use.

Based on these observations, we develop simple energy models by per-

	Intel	Atheros	Phone
A	$0.24 \times n_{tx} + 0.425 \times MIMO + 1.02$	$0.38 \times n_{tx} + 0.108$	1.53
B	$0.045 \times n_{tx} + 0.108$	$0.040 \times n_{tx} + 0.062$	0.036
C	$0.30 \times n_{rx} + 0.61$	$0.142 \times n_{rx} + 0.30$	1.23
D	$0.064 \times n_{rx} + 0.167$	$0.048 \times n_{rx} + 0.106$	0.002

Table 3.1: Parameters in the energy models.

forming least-square fitting to find the coefficients that best match the energy consumption of the different cards. The energy models are as follow:

$$E_{tx} = A \times ETT + B \tag{3.1.1}$$

$$E_{rx} = C \times ETT + D \tag{3.1.2}$$

where the parameters in the models A, B, C, D vary across different wireless cards and are shown in Table 3.1.

We make several observations. First, the energy consumption is a linear function of ETT, as mentioned earlier.

The slope depends on the number of transmitting or receiving antennas. This is intuitive since using more antennas consumes more energy and the amount of extra energy that is consumed relates to how long the antennas are used. The y-intercept of the linear function reflects a constant processing cost for each frame regardless of their duration. Second, the exact parameters across different cards are similar but not identical. For example, the Intel transmitter requires an additional parameter $MIMO$, which indicates whether MIMO mode is enabled. This is a well documented anomaly of the Intel card, where two antennas turn on almost all the hardware required for three antennas, with only 5% energy difference between two and three antennae configurations. This is also reported in [37]. The model for the phone is similar in spirit to the other cards. But since we do not have a smartphone with an embedded MIMO enabled WiFi card, we cannot separate which parts in A and B are from n_{tx} and n_{rx}. The values for the phone are higher than those of the other two cards under 1 antenna because the measured energy from the phone includes everything, such as display, CPU, as well as wireless cards.

Third, the energy consumption depends on the number of antennas, but not the number of streams. For example, as shown in Figure 3.3(a), the energy consumption under 3 antennas using 1, 2, and 3 streams are identical and overlap; similarly for 2 antennas using 1 and 2 streams. Finally, we note that our receiver energy model is conservative (*i.e.*, it may sometimes over-estimate the energy consumption). This is because depending on where the reception fails (*e.g.*, if preamble detection fails, the receiver will stop further processing the signals and the energy consumption is likely to be lower than that of a successful reception). We conservatively assume every transmissions (regardless failures or success) consumes the same amount of receiving energy. Since preambles are quite reliable compared to data symbols, which may be sent at a higher data rate, the approximation error is likely to be small.

Table 3.2 shows mean absolute percentage error (MAPE) of our energy

Card	transmission	reception
Atheros	3.4%	1.3%
Intel	0.65%	1.4%
Phone	4.9%	3.6%

Table 3.2: Mean absolute percentage error of energy models.

models versus the measurement data, defined as

$$MAPE = mean(|\frac{x - x'}{x}|),$$

where x and x' are the actual and estimated energy consumption, respectively. As we can see, the error is consistently below 5%, indicating a close match.

3.2 Rate Adaptation

In this section, we develop an energy aware rate adaptation protocol based on the energy models. Our goal is to select the data rate for the next transmission in order to minimize the energy consumption. In IEEE 802.11n, the data rate is defined as Modulation and Coding Scheme (MCS), which specifies the modulation, FEC coding, and antenna configuration. To achieve this goal, the protocol first obtains Channel State Information (CSI) seen by the receiver, then computes the delivery ratio and energy consumption under different MCS, and selects the MCS that yields the lowest estimated energy. Below we describe each step in detail.

3.2.1 Channel State Information

IEEE 802.11n standard specifies how to calculate and report CSI. The CSI values are a collection of $M \times N$ matrices H_s, each of which specifies amplitude and phase between pairs of N transmit and M receive antennas on subcarrier s. SNR and amplitude A have the following relationship: $SNR = 10log_{10}(A^2/N)$, where N denotes the average power of white noise. For example, Intel WiFi Link 5300 (iwl5300) IEEE a/b/g/n wireless network adapters collects the CSI of each frame preamble across all subcarriers for up to three antennas.

Using the CSI values, we calculate the post-processed SNR (pp-SNR) values for each subcarrier under every supported transmission configuration. The post-processed SNR is the SNR value obtained after MIMO decoding. In MIMO, since a transmitted symbol is received on multiple antennas, the final SNR experienced by the symbol is the combination of the multiple receptions and the combined SNR dictates whether it will be decoded correctly. For spatial multiplexing modes, we use a Minimum Mean Squared Error (MMSE) equalizer to calculate the post-processed SNR. The SNR value for the m^{th} stream on subcarrier s after MMSE equalization can be written as:

$$SNR_m^{MMSE} = \frac{E_s}{N_t N_0} \frac{1}{\left[H^H H + \left(\frac{E_s}{N_t N_0} \right)^{-1} I \right]^{-1}_{m,m}} \tag{3.2.1}$$

where E_s is the total transmission energy across all transmit antennas, N_t is the number of transmit antennas, N_0 is the noise power, H is the channel matrix for subcarrier s (H_{ij} is the channel coefficient of the j-th transmitting antenna to i-th receiving antenna), I is an identity matrix, and H^H is the Hermitian transpose of H matrix. The pp-SNR expression in equation 3.2.1 is applicable for all cases, including when the number of spatial streams is equal to the number of transmit antennas ($N_{ss} = N_t$) and when the number of transmit antennas is less than or equal to the number of receive antennas ($N_t \leq N_r$). Hence, equation 3.2.1 is used for all receive diversity cases since $N_t < N_r$ is for receive diversity.

The calculation of pp-SNR for transmit diversity modes depends on the mechanism used to achieve diversity. The two supported mechanisms in IEEE 802.11n are Space Time Block Coding (STBC) and Cyclic Delay Diversity (CDD). For STBC, which provides full diversity, the pp-SNR can be calculated as:

$$SNR^{STBC} = \frac{E_s}{N_t N_0} \sum_{i=1}^{N_r} \sum_{j=1}^{N_t} |h_{ij}|^2 \tag{3.2.2}$$

where h_{ij} is the channel coefficient of the j-th transmitting antenna to i receiving antenna, N_t and N_r are the numbers of transmit and receive antennas, respectively.

For CDD modes, the SNR can be estimated by [20]:

$$SNR_s^{CDD} = \frac{E_s}{N_t N_0} \sum_{i=1}^{N_r} \left| \sum_{k=1}^{N_t} h_{ik} e^{-j \frac{2\pi s}{N_{fft}} \delta_{cy(k)}} \right|^2 \tag{3.2.3}$$

where $\delta_{cy(k)}$ is the delay defined by the IEEE 802.11n standard for cyclic delay transmission for transmit antenna k. N_{fft} is the FFT size, and s is the subcarrier index. It should be noted that Equation 3.2.3 depends on the subcarrier index.

modulation	BER
BPSK	$Q(\sqrt{2snr})$
QPSK	$Q(\sqrt{snr})$
QAM-16	$\frac{3}{4}Q(\sqrt{snr/5})$
QAM-64	$\frac{7}{12}Q(\sqrt{snr/21})$

Table 3.3: BER for different modulations as a function of SNR

3.2.2 Computing Loss Rate

To compute the loss rate, we first map the pp-SNR of each subcarrier to the uncoded BER using the well-known relationship between SNR and BER as shown in Table 5.1. Then to take into account the frequency diversity (*i.e.*, SNR varies across different subcarriers), as [38] suggests, we compute average BER across all the subcarriers. Next we derive the BER after FEC coding using the error-probability upper bound defined for the Viterbi decoder to map the uncoded BER to coded BER. The Viterbi decoder's probability of bit error is upper bounded as follows according to [80]:

$$BER_{coded}(\rho) = \sum_{d=d_{free}}^{\infty} a_d . P_d(\rho) \tag{3.2.4}$$

$$P_d(\rho) = \begin{cases} \sum_{k=(d+1)/2}^{d} \binom{d}{k}.\rho^k.(1-\rho)^{d-k}, & \text{if d is odd} \\ \frac{1}{2}.\binom{d}{d/2}.\rho^{d/2}.(1-\rho)^{d/2} & \\ + \sum_{k=d/2+1}^{d} \binom{d}{k}.\rho^k.(1-\rho)^{d-k}, & \text{if d is even} \end{cases} \tag{3.2.5}$$

where ρ is the uncoded BER, d_{free} is the minimal hamming distance between two coded sequences, and a_d is the number of incorrect paths of hamming distance d that diverge from the correct path and then re-merge sometime later [35]. The coded BER value can then be used to approximate the frame error rate (FER) as $1-(1-BER_{coded})^L$ assuming independent bit error rate, where L is the frame size.

To further enhance performance, Partial Packet Recovery (PPR) [50] is proposed to let a receiver extract correct bits from a partially corrupted frame. When PPR is used, our goal is to maximize the expected number of delivered bits, which can be computed as $(1-HeaderLoss)(1-BER_{uncoded}) \times L'$, where $HeaderLoss$ is the loss rate of the frame header, L' is the payload size, and $BER_{uncoded}$ is uncoded BER. $BER_{uncoded}$ is used since the FEC is no longer useful for a corrupted frame.

3.2.3 Estimating Energy Consumption

To accurately estimate the energy consumption, an AP or a back-end server should keep a table of the energy models for commonly used WiFi cards. Whenever a new client arrives, it checks the make and model of the wireless card based on either explicit feedback or passive detection of 802.11 wireless drivers[28] or fingerprinting techniques[72] using 802.11 protocol fields. For example, “more fragments”, “retry”, or “power management” bits in the protocol field reveals the wireless card information. Then it computes ETT based on frame loss rate and applies the corresponding energy model to derive the energy consumption for the next transmission under different MCS. When a client’s wireless card has unknown energy profile, it is possible to infer the energy model based on data transmissions. For example, the AP can let the client report the energy consumption at a few data rates under different numbers of antennas to estimate the slope in the energy model. The model is then inserted to the table and can be updated as more measurements become available. As part of our future work, we plan to investigate how quickly we can infer the energy model using such online measurement.

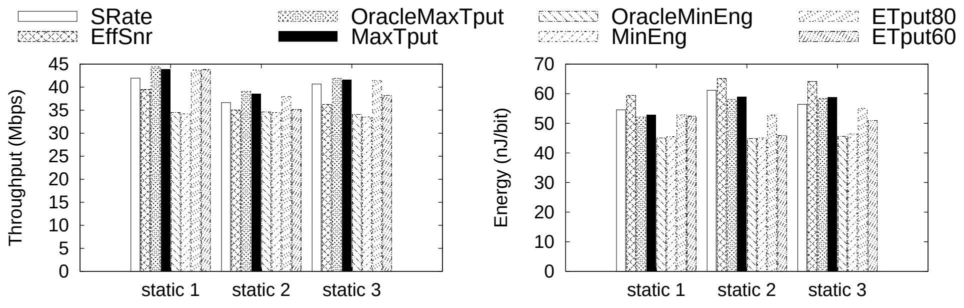
3.2.4 MCS and Antenna Selection

Based on the frame error rate calculated for all MCS, we identify the MCS that have a reasonable delivery rate (*e.g.*, 90% or above). Among these MCS, we select the MCS that yields the minimum energy. Note that we can easily incorporate different objectives in this process, such as minimizing energy or minimizing energy subject to throughput constraint (*e.g.*, throughput is within $X\%$ from the optimal throughput, where X is a configurable knob), or other combinations of throughput and energy. In our evaluation, we also consider several variants that jointly optimize energy and throughput.

3.3 Evaluation

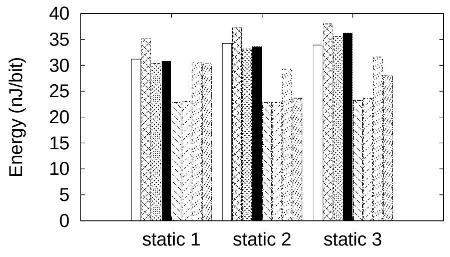
3.3.1 Simulation

We first evaluate various rate adaptation schemes using trace-driven simulation. We quantify the performance of different schemes in terms of their energy consumption and throughput.



(a) Transmitter throughput

(b) Intel transmitter energy



(c) Atheros transmitter energy

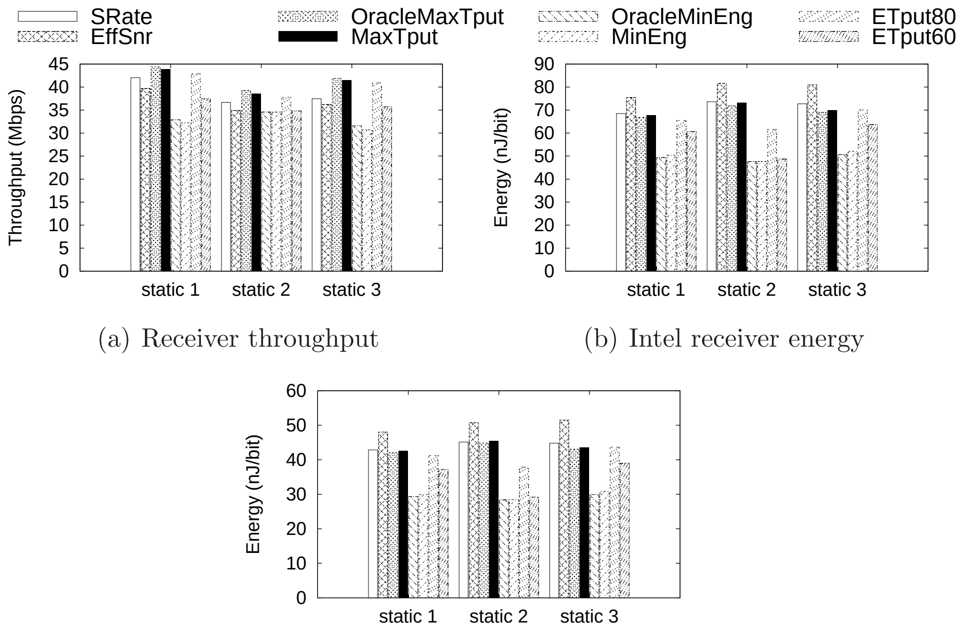
Figure 3.4: Transmitter Energy comparison in static networks.

3.3.1.1 Simulation Methodology

We develop a simulator in python using the CSI traces. For each frame, the data rate is selected according to different rate adaption schemes. Then we determine if the frame is successfully received using pp-SNR and taking into account FEC. The simulator also supports Partial Packet Recovery (PPR), which uses uncoded BER to determine the number of bits correctly received.

We compare the following rate adaptation schemes:

- **Sample Rate (SRate):** Sample Rate [10] is a widely used rate adaptation scheme. It probes the network at a random rate every 10 frames and selects the rate that minimizes transmission time including retransmission time.



(a) Receiver throughput

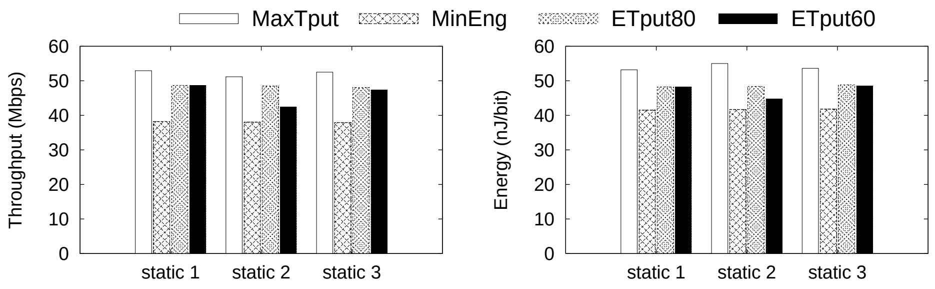
(b) Intel receiver energy

(c) Atheros receiver energy

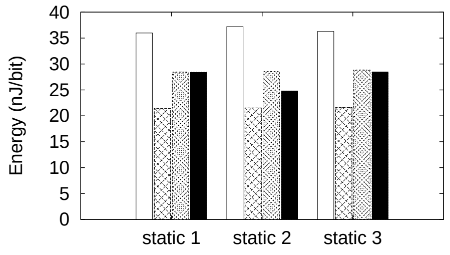
Figure 3.5: Receiver Energy comparison in static networks.

Its goal is to maximize throughput without considering energy consumption. We implement an extended version of Sample Rate which supports MIMO transmission modes. The original Sample Rate starts at the highest rate and reduces the rate based on channel conditions. We extend this idea and start at the highest rate using all antennas and then reduce or increase the MCS or the number of antennas based on throughput of the previous transmissions.

- **Effective SNR (EffSNR):** [38] proposes selecting the data rate based on effective SNR derived from the CSI values. It computes the post-processed SNR for each subcarrier and maps it to BER. Then it calculates the average



(a) Transmitter throughput (b) Intel transmitter energy



(c) Atheros transmitter energy

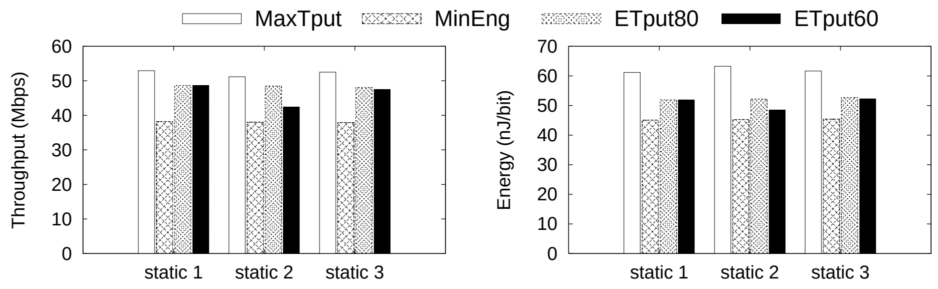
Figure 3.6: Transmitter Energy comparison in static networks using PPR.

BER across all subcarriers and converts the average BER to effective SNR with the same BER. Effective SNR also aims to maximize throughput and does not consider energy consumption.

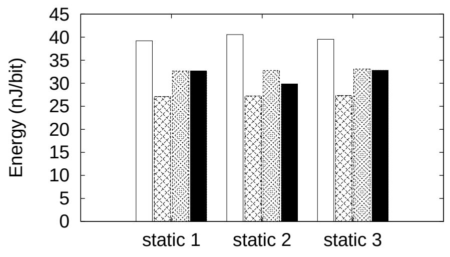
- **Maximum Throughput (MaxTput):** Maximum Throughput rate adaptation uses the rate selection scheme in Section 3.2. Unlike energy minimization scheme, it picks the MCS that maximizes throughput.

- **Minimum Energy (MinEng):** Minimum Energy is our proposed rate adaptation scheme from Section 3.2. It picks the MCS that minimizes the energy consumption while ensuring the frame delivery rate is above 90%.

- **Minimum Energy with Throughput Constraint (ETput$X$):** This



(a) Receiver throughput (b) Intel receiver energy

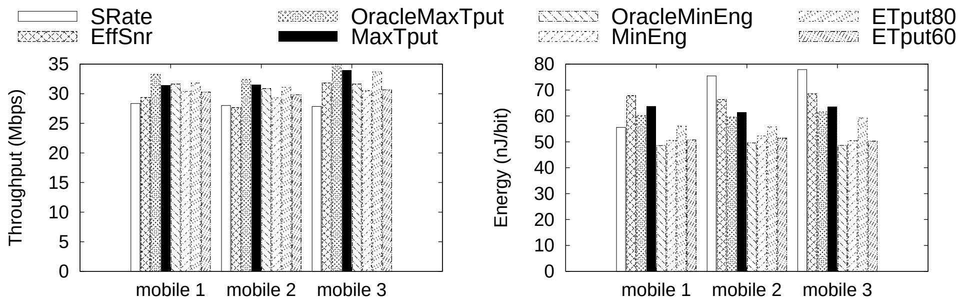


(c) Atheros receiver energy

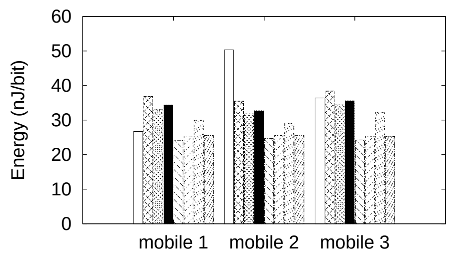
Figure 3.7: Receiver Energy comparison in static networks using PPR.

scheme aims to select the MCS that minimizes the energy provided the throughput is no less than $X\%$ of the maximum throughput. We vary X to yield different variants. For example, ETput80 means minimizing energy while ensuring throughput is at least 80% of the maximum throughput.

The energy consumption is derived using the energy models for Intel and Atheros as described in Section 3.1.2. We collect three channel traces from static environments, and another three traces from mobile environments with human walking speed. The three mobile traces are collected in an office environment using 1 moving receiver and 3 static senders. The three static senders are 7m away from each other. Each trace corresponds to one of the three senders transmitting while the receiver is moved at a walking speed.



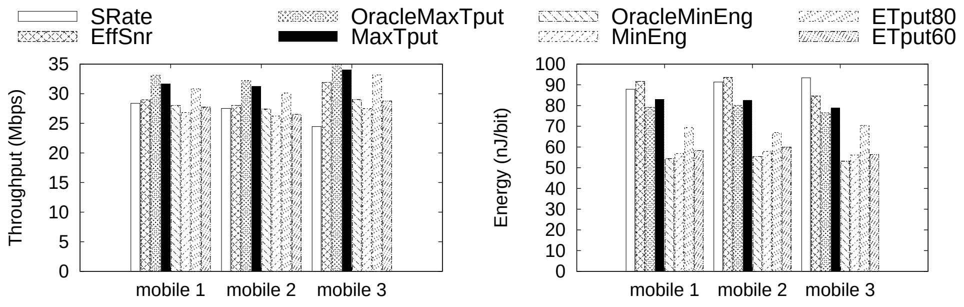
(a) Transmitter throughput (b) Intel transmitter energy



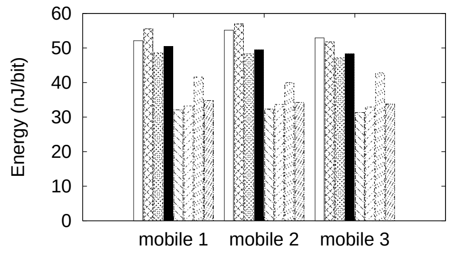
(c) Atheros transmitter energy

Figure 3.8: Transmitter Energy comparison in mobile networks.

We use Intel WiFi Link 5300 (iwl5300) IEEE a/b/g/n wireless network adapters to collect the CSI of each frame preamble across all subcarriers. These NICs have three antennas. We enable all three antennas at both the sender and receiver. The modified driver [39] reports the channel matrices for 30 subcarrier groups, which is about one group for every two subcarriers in a 20 MHz channel according to the standard [1] (*i.e.*, 4 groups have one subcarrier each, and the other 26 groups have two subcarriers each). We use 1000-byte packets and MCS-16, with a transmission power of 15 dBm. MCS-16 has 3 streams, so the NICs report CSI in the form of 3×3 matrices for each frame.



(a) Receiver throughput (b) Intel receiver energy



(c) Atheros receiver energy

Figure 3.9: Receiver Energy comparison in mobile networks.

3.3.1.2 Simulation Results

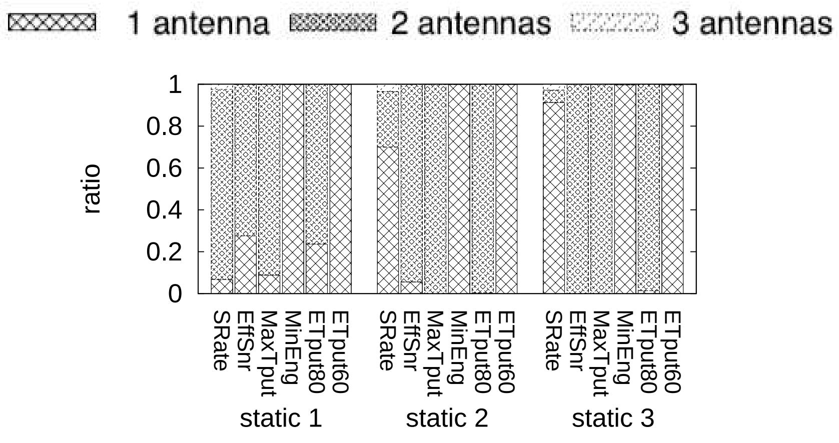
Static networks: First, we evaluate the performance in static networks using three traces collected in a static environment. Each trace contains 2000 CSI samples. Figure 3.4 plots the throughput and energy consumption for the transmitter. As we can see, compared to the scheme that maximizes throughput, the energy-aware rate adaptation scheme consumes 14-24% less energy for the Intel card and 25-35% less energy for the Atheros card. The throughput loss for both cards is 10-22%.

Compared with Effective SNR and Sample Rate, minimum energy reduces transmitter energy by 17-31% for the Intel card and 26-39% for the

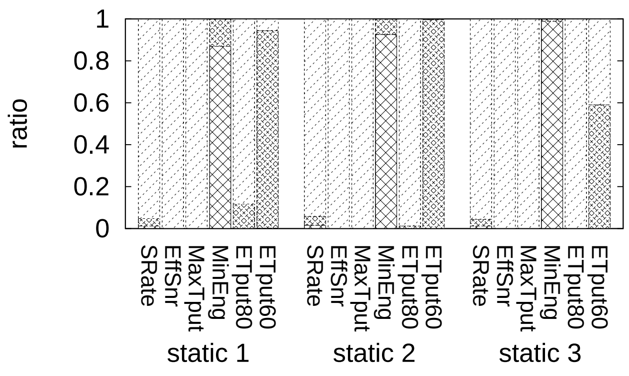
Atheros card while the throughput loss is 1-19%. The energy saving is higher and throughput reduction is lower in the latter cases because Effective SNR or Sample Rate are not optimal for either throughput or energy. ETput$X$ balances the throughput and energy. For example, compared with the maximum throughput scheme, ETput80, which minimizes energy while ensuring at least 80% of the maximum throughput, saves energy of up to 10% and 13% for the Intel and Atheros transmitters, respectively, while reducing throughput within 1%. Moreover, OracleMinEng and OracleMaxTput know the exact CSI of the next frame and eliminate the performance degradation caused by prediction error. As we can see, the CSI prediction error causes only 1-2% more energy consumption and 1-2% throughput reduction, indicating the impact of prediction error is small.

Figure 3.5 shows the performance results for the receiver. Compared with the scheme that maximizes throughput, the energy-aware rate adaptation scheme reduces the receiver's energy by 25-35% for the Intel card and 30-37% for the Atheros card at the cost of 10-26% throughput reduction.

Compared with Effective SNR and Sample Rate, minimum energy reduces receiver energy by 26-42% for the Intel card and 30-44% for the Atheros card while the throughput loss is 1-23%. As before, ETputX balances energy and throughput: ETput80 reduces energy by 10% and 13% for the Intel and Atheros receivers, respectively, with almost no throughput loss. In addition, compared with OracleMinEng and OracleMaxTput, MinEng incurs only 1-4% more energy and 1-5% throughput loss.



(a) # tx antennas used for Intel transmitter



(b) # rx antennas used for Intel receiver

Figure 3.10: Number of antenna used in static networks.

Figure 3.10 shows the number of antennas used by each scheme. We can see that the energy-aware rate adaptation tends to use one antenna to minimize energy consumption. Meanwhile, it also uses two antennas in some cases whenever the reduced transmission time can offset the additional energy required by an extra antenna. The maximum throughput scheme, on the other hand, does not care about the energy consumption and uses as many antennas as possible to achieve better throughput. ETput$X$ schemes try to balance MinEng and MaxTput schemes and the number of antennas they use is between those used by the two schemes.

We also ran simulations using a Partial Packet Recovery(PPR). As shown in Figure 3.6, in this case the energy-aware rate adaptation reduces the transmission energy by 22-24% for the Intel card and by 40-42% for the Atheros card. These energy savings are achieved at the cost of 26-28% throughput reduction for both cards.

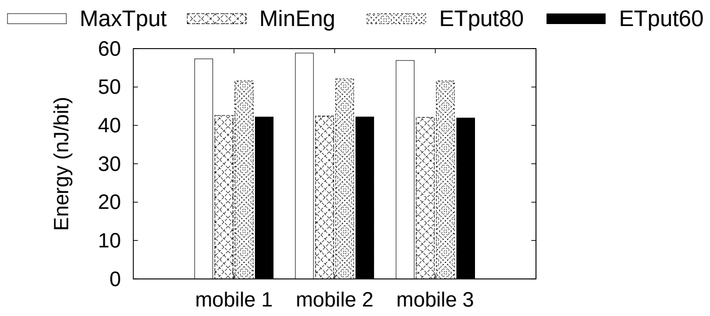
As shown in Figure 3.7, the energy savings for the PPR receiver are 26-28% and 31-33% for the Intel and Atheros cards, respectively. The throughput loss for these cards is 26-28%. To trade off between throughput and energy savings, ETput80 saves energy by 9% and 21% for Intel and Atheros, respectively. The throughput reduction is within 9%. Moreover, comparing PPR energy saving with non PPR energy savings, we see PPR based scheme improves the energy by 6-23% by extracting correct symbols from partially corrupted frames.

Mobile networks: Next we evaluate the different schemes using the three mobile traces. Figure 3.8 and 3.9 summarize the results.

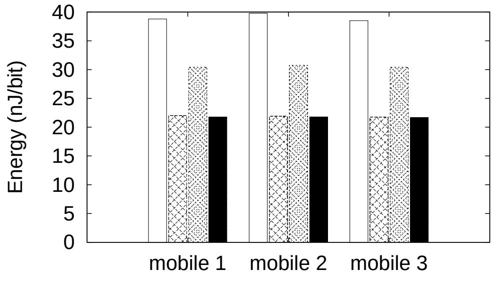
Compared with the scheme that maximizes throughput, minimum energy reduces transmitter energy by 15-21% for the Intel card and 22-29% for the Atheros card. For both Intel and Atheros, the throughput loss is 3-10%. Compared with Effective SNR and Sample Rate scheme, minimum energy reduces transmitter energy by 9-35% for the Intel card and 5-49% for the Atheros card. The throughput of minimum energy is higher than Effective SNR and Sample Rate in some mobile traces since the latter two are not optimal for throughput.

For the receiver, minimum energy reduces energy by 29-31% for the Intel card and 32-34% for the Atheros card while reducing the throughput by 15-19% compared to maximum throughput scheme. Compared with Effective SNR and Sample Rate scheme, minimum energy reduces receiver energy by 34-40% for the Intel card and 36-41% for the Atheros card. To trade off between throughput and energy savings, ETput80 scheme reduces the throughput by 2% compared to maximum throughput scheme while providing energy savings of 16% and 18% for Intel and Atheros receivers, respectively. Compared with OracleMinEng and OracleMaxTput, the CSI prediction error causes only 2-6% more energy consumption and 3-6% throughput reduction. The degradation in mobile traces is slightly larger than that in static traces as expected since the channel variation in mobile traces increases the CSI prediction error. Nevertheless, the degradation in this case is still small. As in the static networks, the energy-aware rate adaptation uses one antenna in most cases, and uses more antennas to reduce transmission time if possible. The maximum throughput scheme uses as many antennas as the channel condition allows.

Figure 3.11 and 3.12 further show the performance of various PPR versions of rate adaptation schemes. In this case, the minimum energy scheme reduces Intel transmitter energy by 26-28% and Atheros energy by 43-45%. The throughput loss is 22-24%. For receiver, the energy savings for Intel are



(a) Intel transmitter energy

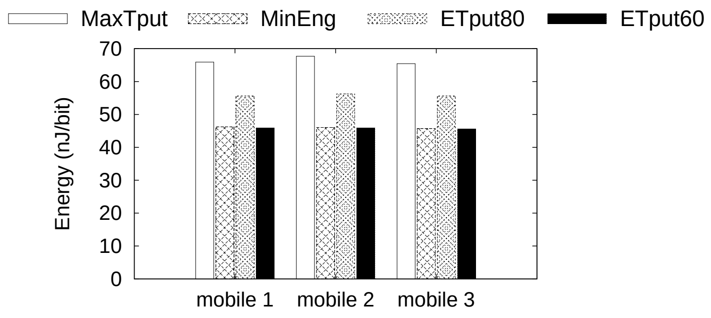


(b) Atheros transmitter energy

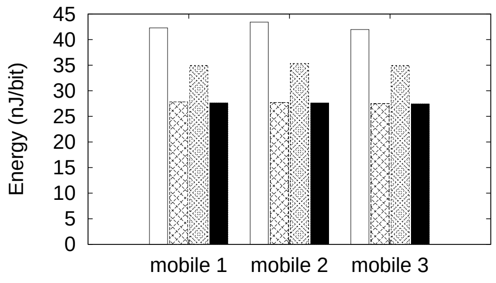
Figure 3.11: Transmitter Energy comparison in mobile networks using PPR.

30-32% and for Atheros are 34-36%. The throughput loss is 22-24%. Compared with non-PPR counterparts, the PPR versions lead to 13-20% energy savings. To balance the throughput and energy savings, ETput80 scheme reduces the throughput by 8% while providing energy savings of 10% and 22% for Intel and Atheros transmitters, respectively.

Impact of frame sizes: In order to take full advantage of the high data rates offered by IEEE 802.11n, using large frames is strongly recommended. Therefore, we further evaluate the impact of frame sizes. Figure 3.13 shows the number of antennas selected by MinEng for the Intel card as we vary the frame sizes from 1000 bytes to 5000 bytes. As we can see, MinEng always



(a) Intel receiver energy

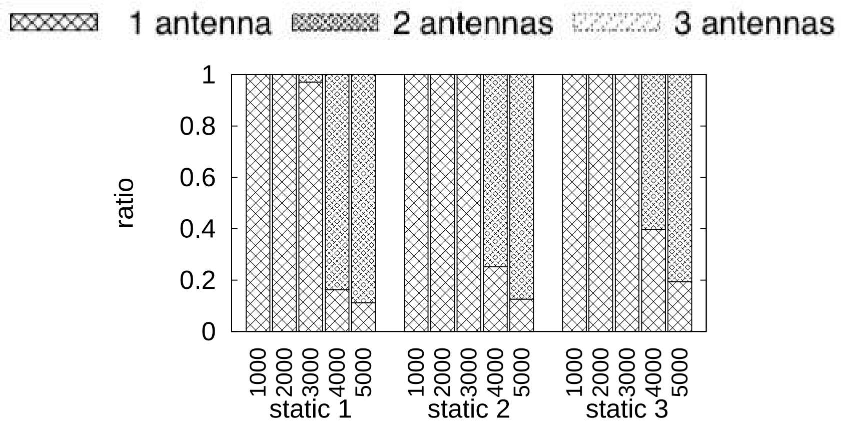


(b) Atheros receiver energy

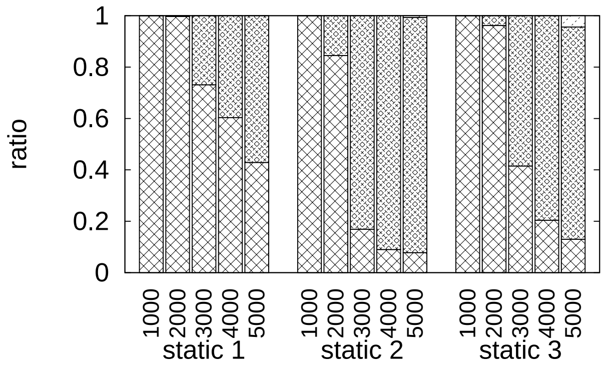
Figure 3.12: Receiver Energy comparison in mobile networks using PPR.

selects the one antenna rate for 1000-byte frames in our traces. However, as the frame size increases, we see more transmissions use multiple antennas. For 5000-byte frames almost all transmissions use two antennas. This indicates as frame size increases, it becomes more advantageous to use multiple antenna rates to minimize energy.

Multiple antennas provide energy saving for larger frames because for small frames the preamble transmission time dominates the total transmission time. Hence, using multiple antennas only results in small reduction in ETT, which does not offset the additional energy required to power up multiple antennas. As the frame size increases, using multiple antennas leads to larger



(a) # tx antennas used for Intel transmitter



(b) # rx antennas used for Intel receiver

Figure 3.13: Number of antennas used for different frame sizes.

reduction in ETT, which more than offsets the additional energy required to power up more antennas.

Other energy objectives: Our scheme is general and can easily support other energy objectives. To give another example, here we consider minimizing the total energy consumption from both sender and receiver, which is especially interesting in ad-hoc networks where the sender and receiver are both mobile nodes with limited energy.

Figure 3.14 shows the performance of MaxTput and MinEng scheme with different objectives in static traces. The performance of mobile traces is similar and omitted for brevity. As it shows, MinEng leads to 19-30% total energy saving with 10-26% throughput reduction. ETput80 balances the total energy consumption and throughput, and reduces energy by 1-13% at a 1-2% throughput loss. ETput60 reduces the total energy by 2-28% with a 5-9% throughput reduction.

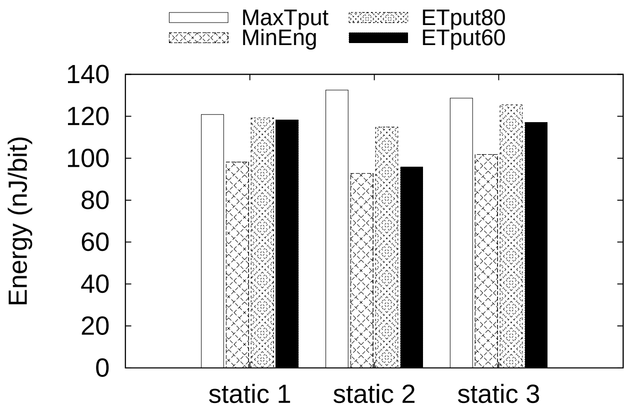


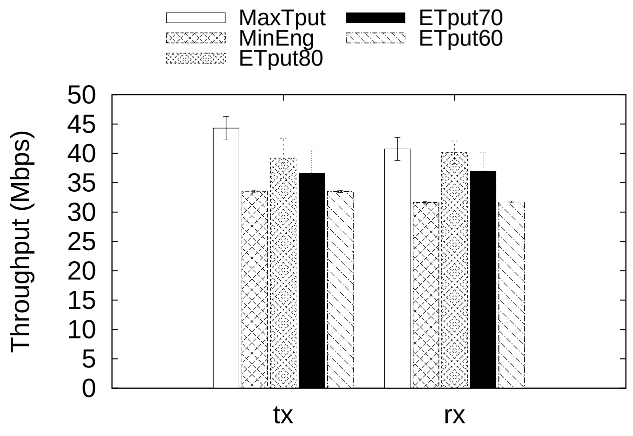
Figure 3.14: Comparing total energy consumption in static networks.

3.3.2 Testbed Evaluation

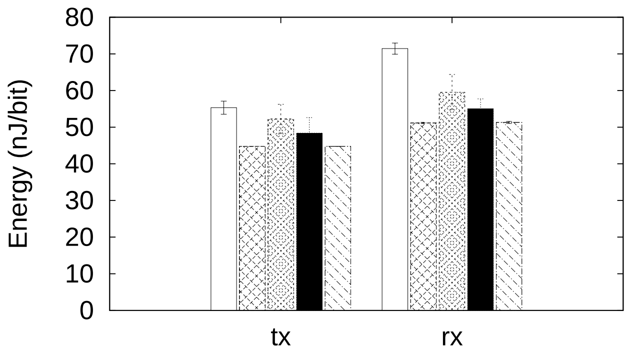
3.3.2.1 Implementation

We implement different rate adaptation schemes in the Intel WiFi link 5300 driver. We use the tool in [38] to extract CSI from the Intel card at the receiver. The receiver uses the extracted CSI information to calculate the throughput and energy consumption for each MCS. The receiver then uses these calculated values to select the appropriate MCS and informs the transmitter to use the selected MCS.

We conduct testbed experiments using two desktop machines. For each experiment, we send 200 UDP packets with 1000-byte payload. The experiments are conducted in static and mobile scenarios. For mobile experiments, initially the machines are placed close to one another and then the receiver is moved away from the transmitter at a walking speed. For each configuration, we report the average throughput and energy consumption across 10 runs for static experiments and across 5 runs for mobile experiments.



(a) Throughput of the static trace in testbed



(b) Energy of the static trace in testbed

Figure 3.15: Comparison of performance of the static trace in the testbed.

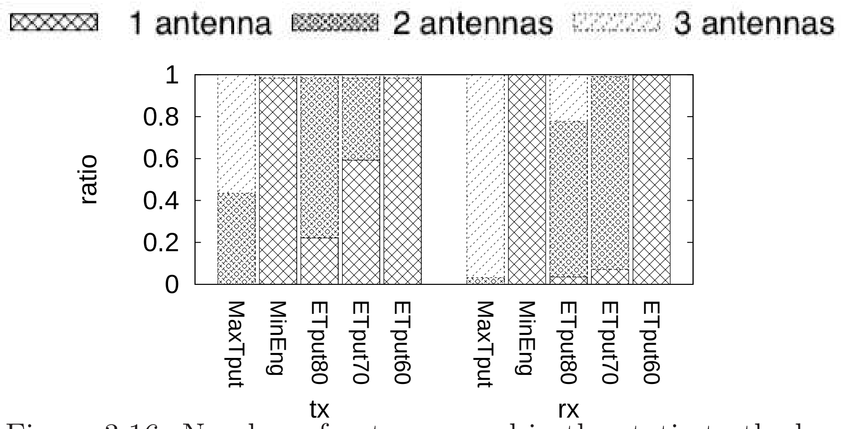


Figure 3.16: Number of antenna used in the static testbed.

3.3.2.2 Testbed Results

Figure 3.15 shows the throughput and energy consumption for static experiments. As we can see, MinEng reduces the energy consumption by 19% for the transmitter and by 28% for the receiver. The throughput reduction is 24% for the transmitter and 22% for the receiver. ETputX smoothly trades-off between the two objectives. For example, ETput80 reduces energy by 6% at a 11% throughput loss for the transmitter. For the receiver, ETput80 reduces the energy by 16% with a throughput reduction of 2%. Figure 3.16 shows the number of transmit and receive antennas used during the experiment. Due to the static channel, the schemes use the same MCS for most transmissions which is expected. MinEng uses a single antenna at both the transmitter and receiver to reduce energy. In comparison, MaxTput utilizes two and three antennas to achieve higher throughput at the cost of additional energy.

Figure 3.17 shows how MCS changes over an mobile experiment for MaxTput and MinEng. MCS 0 to 7 use 1 antenna, MCS 8 to 15 use 2 antennas, and MCS 16 to 23 use 3 antennas. In each case, the number of spatial streams is equal to the number of antennas. In region 1, when the channel is good, MaxTput transmits using all 3 antennas at MCS 22. Since MinEng tries to minimize energy, it uses MCS 6, the highest 1-antenna rate that can be supported by the current channel. MinEng saves 16.9% energy over MaxTput in this region. As the receiver moves away from the transmitter, the channel condition degrades and forces MaxTput to drop to MCS 14, while MinEng continues to use MCS 6. The energy improvement reduces to 11.9% because MCS 14 used by MaxTput consumes less energy than its previous MCS 22 due to a fewer number of antennas used. In region 3, MaxTput drops from MCS 14 to MCS 12. Since MCS 12 still uses 2 antennas but takes longer to transmit than MCS 14, MCS 12 consumes 15.5% more energy than MCS 14. In comparison, MinEng continues to use MCS 6 and its energy saving jumps to 21%. In region 4, the MinEng drops to MCS 5, resulting in longer transmission time. Since MaxTput still uses MCS 12, the energy saving of MinEng reduces to 20.06%. It is interesting to note that even though the channel degrades continuously, the energy savings do not follow the trend. In fact, region 2 has the least gap between MaxTput and MinEng while region 3 has the highest. In all cases, MinEng yields significant energy savings.

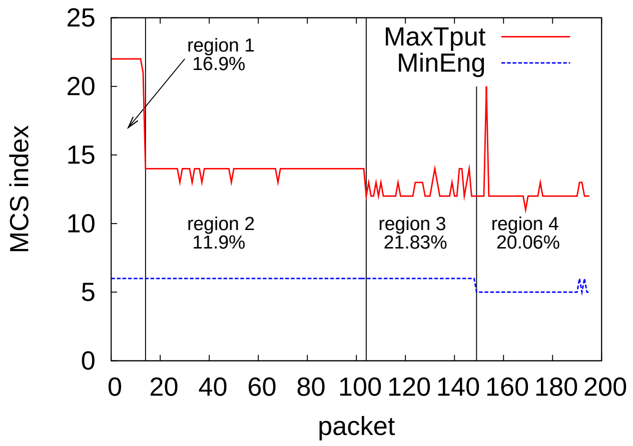


Figure 3.17: The evolution of MCS over time for MinEng and MaxTput in a mobile experiment.

Chapter 4

Delivery Ratio Estimation

4.1 Burstiness in Wireless Channel

4.1.1 Background

In OFDM, a channel is divided into multiple orthogonal subcarriers, and the data is spread on to these subcarriers for simultaneous transmission. Due to frequency selective fading and narrow-band interference, the SNR of these subcarriers vary across different frequencies.

The IEEE 802.11n standard specifies how to measure and report channel state information (CSI). CSI is essentially a collection of $K_t \times K_r$ matrices H_s, each of which specifies amplitude and phase between pairs of K_t transmitting antennas and K_r receiving antennas on subcarrier s. SNR relates with amplitude A as follows: $SNR = 10log_{10}(A^2/N)$, where N denotes the average power of white noise. Following the IEEE 802.11n, wireless network adapters (*e.g.*, Intel WiFi Link 5300 (iwl5300)) report the CSI of a preamble in each frame.

4.1.2 WiFi interleaver

According to the IEEE 802.11 standard, all data bits are interleaved by a block interleaver with a block size corresponding to the number of bits in one

OFDM symbol [1]. For simplicity, we consider interleaving in a single antenna case for illustration, and a similar performance issue exists in MIMO cases. Let N_{CBPS} denote the block size, and N_{BPSC} denote the number of coded bits per subcarrier. The block interleaver is a two-step permutation procedure. The first permutation step maps adjacent coded bits to non-adjacent subcarriers. The second permutation maps adjacent coded bits alternatively to less and more significant bits of the constellation to avoid long runs of low reliability bits. Let k denote the index of the coded bit before the first permutation, i denote the index after the first permutation, and j denote the index after the second permutation. The two permutation steps are defined as follows, where $k = 0, 1, ..., N_{CBPS} - 1$, $i = 0, 1, ..., N_{CBPS} - 1$, and s is determined by N_{BPSC} according to $s = max(N_{BPSC}/2, 1)$.

$$i = (N_{CBPS}/13)(k \ mod \ 13) + floor(k/13) \tag{4.1.1}$$

$$j = s \times floor(i/s) + (i + N_{CBPS} - floor(13 \times i/N_{CBPS})) \ mod \ s \tag{4.1.2}$$

Figure 4.1 shows the standard interleaver for BPSK. IEEE 802.11n has 52 data subcarriers, so there are 52 BPSK symbols in each OFDM symbol. The interleaver stripes 52 bits by placing them in a 4×13 grid row-wise and reading them column-wise so that adjacent bits are spread 4 subcarriers apart. It is quite likely SNR at these subcarriers still experience similar performance, thereby resulting in bursty errors. This effect persists under other modulation schemes: the adjacent bits are still 4 subcarriers away and bursty errors still

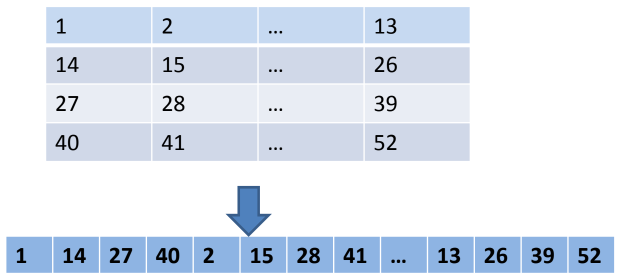
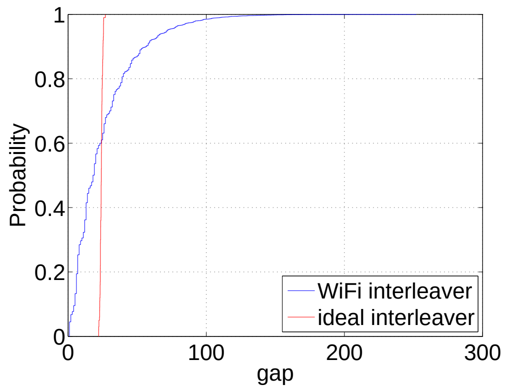


Figure 4.1: The WiFi interleaver for BPSK.

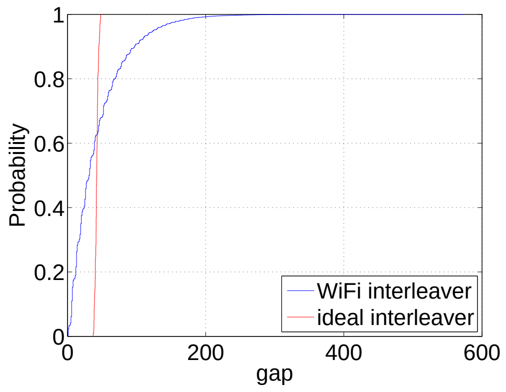
exist.

4.1.3 Trace collection

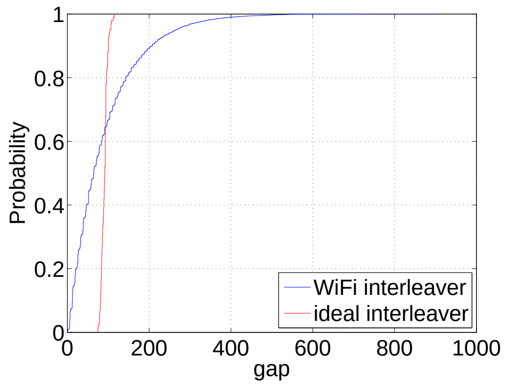
We analyze real WiFi traces to understand their burstiness after interleaving. The traces were collected from WiFi networks using Intel WiFi Link 5300 (iwl5300) wireless network adapters. Three channel traces are from static environments, and another three traces are from mobile environments with human walking speed. The three mobile traces are collected in an office environment using 1 moving receiver and 3 static senders. The three static senders are 7m away from each other. Each trace corresponds to one of the three senders transmitting while the receiver is moved at a walking speed. The NICs have three antennas, which are all enabled in our measurement. The modified driver [39] reports the channel matrices for 30 subcarrier groups, which is about one group for every two subcarriers in a 20 MHz channel ac-



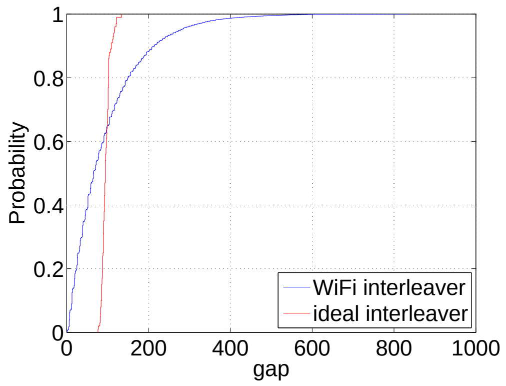
(a) 1/2 FEC



(b) 2/3 FEC



(c) 3/4 FEC



(d) 5/6 FEC

Figure 4.2: CDF of gaps between two consecutive errors in the WiFi interleaver and an ideal interleaver.

.

cording to the standard [1] (*i.e.*, 4 groups have one subcarrier each, and the other 26 groups have two subcarriers each). We use 1000-byte frames, MCS-16, and a transmission power of 15 dBm. MCS-16 has 3 streams, so the NICs report CSI in the form of 3×3 matrices for each frame. So altogether we have traces of 27 static links and 27 mobile links.

4.1.4 Bursty errors

The IEEE 802.11n supports four types of FEC: 1/2 (*i.e.*, half redundancy), 2/3 (*i.e.*, one third redundancy), 3/4, and 5/6. So we focus on these FEC coding rates. Figure 4.2 plots CDF of the gap between two consecutive errors. The gaps in the WiFi interleaver have skewed distribution, with a much larger fraction concentrated on the lower end than the ideal interleaver, which equally spaces the error across a frame. The gap in the ideal interleaver is not a constant due to a varying number of errors in each frame. These results show that the errors are still bursty under the WiFi interleaver.

4.2 Delivery Ratio Estimation Techniques

Motivated by the bursty errors despite the use of the standard WiFi interleaver, in this section we develop two methods to estimate delivery rates and explicitly capture the bursty errors. We focus on convolutional coding, the default used in IEEE 802.11 a/b/g/n/ac. The convolutional code used in WiFi takes data bits as input and generates coded bits that do not include original data bits (*i.e.*, non-systematic coding). Either all bits in a frame are decoded or nothing is decoded.

4.2.1 Method 1: Lookup Table Based Estimation

Our first scheme randomly samples the error patterns based on CSI, data rate, and interleaver. To achieve high efficiency, we build a lookup table that maps an error pattern in a sliding window to a delivery rate and then online computes the frame delivery rate based on lookup results. Below we describe the detailed scheme.

4.2.1.1 Random sampling

The success rate of Viterbi decoding is hard to model analytically (*e.g.*, [93, 101]). Instead, we can empirically compute the delivery rate for a given CSI over many frames, and use the average delivery rate as the estimation. Specifically, we can generate frames with random payload, use the current data rate (*i.e.*, modulation scheme and FEC) and interleaver to map bits onto subcarriers, corrupt the frames according to the CSI, and feed each corrupted frame to the Viterbi decoder to derive the average frame delivery rate after FEC decoding. Figure 4.3 shows the average estimation error across different MCS and channel traces as we vary the number of frames sampled from 10 to 500. As it shows, 100 sampled frames give a good balance between accuracy and efficiency.

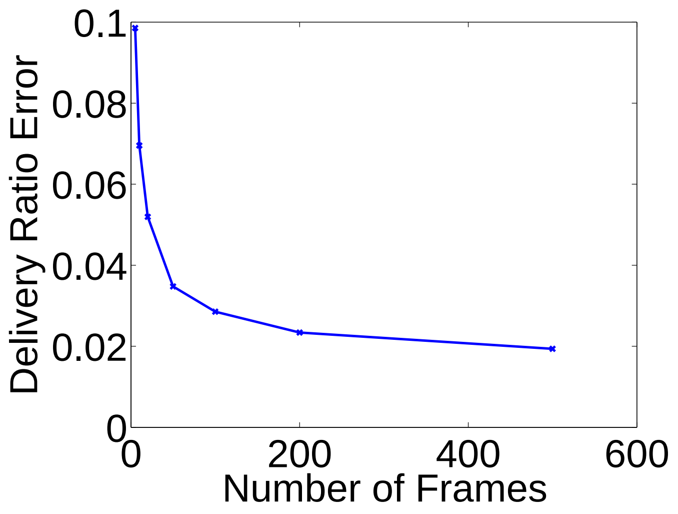


Figure 4.3: Impact of the number of frames sampled.

4.2.1.2 Overview

In order to achieve high accuracy of delivery rate estimation, hundreds of frames are required. It is too expensive to run Viterbi decoding on hundreds of frames online. A natural approach is to run Viterbi decoding offline and build a lookup table in advance. Note that this lookup table takes error patterns as the input, so it is independent of wireless channel or hardware, and we can use the same table across all devices under all channel conditions.

Specifically, the table includes whether decoding is successful for each error pattern (*e.g.*, 0 or 1 sequence where 0 indicates no error in the bit and 1 indicates an error). For example, one entry in the table for a given FEC may have [110000011111000000, fail], which indicates that the frame decoding fails if the bits in the frame (after de-interleaving at the receiver) has the corruption pattern 110000011111000000. Note that we only consider the bit corruption pattern, but not the content of the frame in order to determine whether it can be successfully decoded, because the frame content is not available a priori and the impact of frame content on delivery rate is much less significant than the error patterns.

A major challenge is what should be the index for the lookup table in order to achieve high accuracy and efficiency. Using an error pattern for an entire frame is prohibitively expensive since there are $2^{FrameSize}$ error patterns. We develop a sliding-window-based lookup table approach to enhance efficiency. Specifically, the lookup table is built in advance, and a receiver performs the following steps online:

1. uses the current data rate and interleaver to determine which subcarrier each bit is assigned to;
2. estimates the SNR of each subcarrier using a preamble as usual;
3. determines BER or joint error probability based on the modulation and SNR of the assigned subcarrier;
4. generates random samples of error patterns according to BER;
5. looks up tables to determine the decoding success rate for each error pattern in a sliding window;
6. estimates frame delivery rate based on delivery rates of relevant sliding windows for each sampled frame;
7. estimates the delivery rate as the average frame delivery rate over all randomly sampled frames.

Steps 1), 2), 4) and 7) are standard. Below we elaborate steps 3), 5), and 6).

4.2.1.3 Mapping SNR to BER

Mapping SNR to BER is an important step in many wireless schemes, including effective SNR. We improve the accuracy of existing mapping by capturing that (i) different bit positions in QAM experience different BER under the same SNR and (ii) there is correlation between BER of different bits in the same symbol. This is an important contribution, and useful to many wireless schemes, including other rate adaptation schemes.

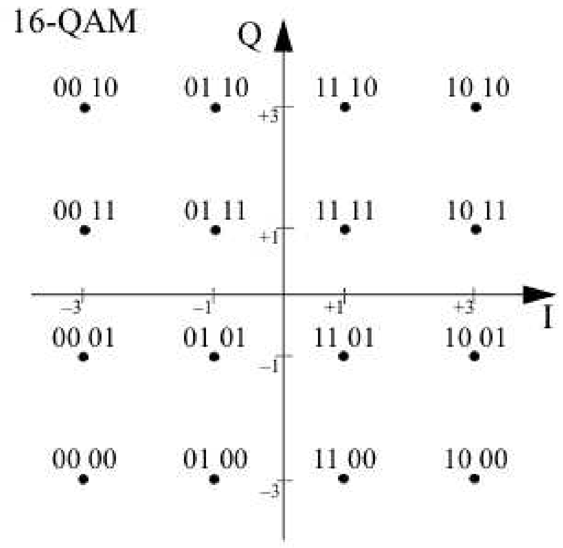
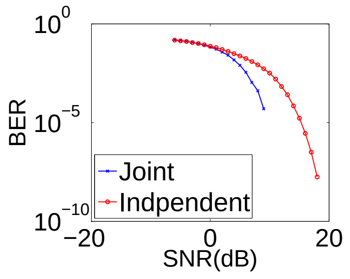
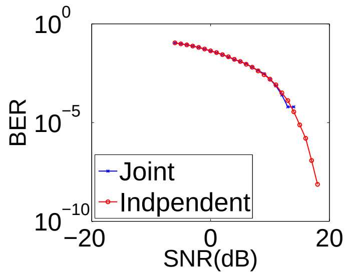


Figure 4.4: Constellation points for QAM-16 with bits $\mathbf{b_1 b_2 b_3 b_4}$

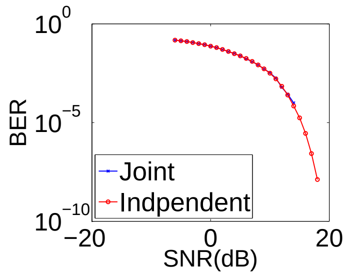
We use the standard formulas to compute BER in BPSK and QPSK: $Q(\sqrt{2snr})$ for BPSK and $Q(\sqrt{snr})$ for QPSK. To compute the BER of QAM (*e.g.*, QAM-16 and QAM-64 used in IEEE 802.11n), we find that the standard formulas do not work for two reasons. First, different bits in QAM may experience different BER. As shown in Figure 4.4, the symbols that differ in bits 1 or 3 have larger minimum distance than those that differ in bits 2 or 4. As a result, BERs of bits 1 and 3 are lower than those of bits 2 and 4. We empirically find that the BERs of bits 1 and 3 match well with $\frac{1}{2}Q(\sqrt{snr/5})$, and BERs of bits 2 and 4 match $Q(\sqrt{snr/5})$. The corresponding functions for QAM-64 are $\frac{1}{4}Q(\sqrt{snr/21})$ for bits 1 and 4, $\frac{1}{2}Q(\sqrt{snr/21})$ for bits 2 and 5, and $Q(\sqrt{snr/21})$ for bits 3 and 6.



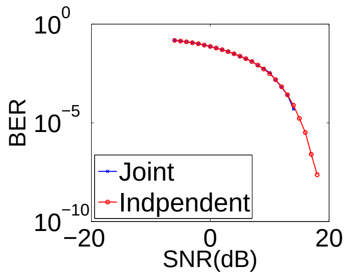
(a) Joint error of bits 1 and 2



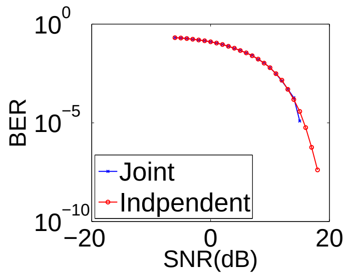
(b) Joint error of bits 1 and 3



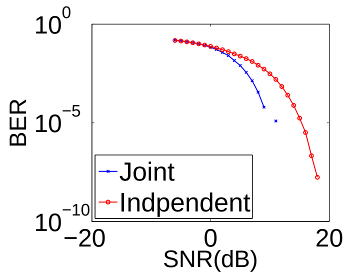
(c) Joint error of bits 1 and 4



(d) Joint error of bits 2 and 3



(e) Joint error of bits 2 and 4



(f) Joint error of bits 3 and 4

Figure 4.5: Joint error probabilities of 2-bits.

Moreover, we find the bits in QAM have considerable correlation. Figure 4.5 compares the joint error probability of two bits from real traces versus estimation based on independence assumption. The estimated joint error deviates from the independence assumption for bits 1, 2 and bits 3, 4, whereas the match is good for the other 2-bit combinations. The gap for bits 1,2 and bits 3,4 indicate negative correlation between these bits. A closer look at Figure 4.4 reveals that if bit 1 is correct, bit 2 is incorrect when the received symbol is over $r/2$ away from the transmitted symbol, where r is the horizontal or vertical separation between two adjacent symbols; if bit 1 is incorrect, bit 2 is incorrect when the received symbol is over r away from the transmitted symbol. The same reasoning applies for bits 3 and 4. Moreover, using similar analysis, we find that the bits 1 and 2 are independent of bits 3 and 4. Based on these observations, we map SNR to joint error probability of bits 1, 2 and bits 3, 4. This is achieved as follow: for each SNR, we generate random symbols, go through modulation and demodulation to determine error patterns, and then compute average probability of error patterns 00, 01, 10, and 11. This yields a lookup table that maps SNR to the probability of having error patterns 00, 01, 10, and 11. We conduct similar analysis for QAM-64, and find bits 1, 2, and 3 are correlated, so are bits 4, 5, and 6, whereas bits 1, 2, 3 are independent of bits 4, 5, and 6. Therefore we generate a similar lookup table that maps SNR to the probability of any 3-bit error pattern.

4.2.1.4 Building a lookup table for a sliding window

Due to finite state space of the convolutional code, the impact of errors does not propagate beyond a certain point. Therefore, we can use a sliding window to capture impact of errors. The sliding window size depends on the type of FEC. We empirically derive the window size for different FEC codes by gradually increasing the window size until a further increase offers little improvement in the delivery rate estimation. We find that the window size is 75 bits for 1/2 FEC, 50 bits for 2/3 and 3/4 FEC, and 40 bits for 5/6 FEC.

Therefore, instead of building a lookup table for entire frames, we build a lookup table for a sliding window with different error patterns.

Pruning the table: Building a table for a sliding window is still too expensive since there are $2^{winSize}$ error patterns, where $winSize$ is the sliding window size. Interestingly, after analyzing the delivery rate of different error patterns, we observe that whenever the number of errors is lower than $lowThresh$ in the window, the Viterbi decoding is always successful; whenever there are more than $highThresh$ errors, the decoding fails almost all the time; only when the number of errors is in between, the decoding error varies according to the error pattern. This is consistent with our expectation of FEC decoding.

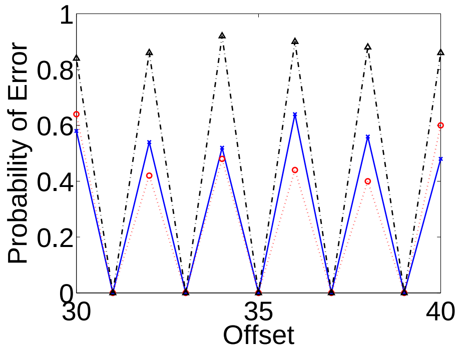
We empirically find $lowThresh$ are 5, 3, 2, and 2 for 1/2, 2/3, 3/4 FEC, and 5/6 FEC, respectively, and the corresponding $highThresh$ are 10, 5, 4 and 4, respectively. This suggests we can build lookup tables for the middle cases (*i.e.*, 5-10 errors for 1/2 FEC, 3-5 errors for 2/3 FEC, and 2-4 errors for both

3/4 FEC and 5/6 FEC).

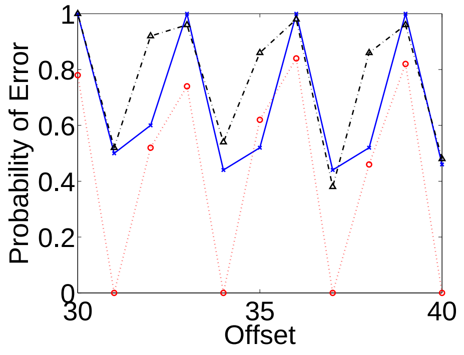
Taking into account position of the sliding window: For the middle cases where the number of errors is not too high or too low, the decoding rate depends on not only the error pattern but also the position of the window. For example, when using 1/2 FEC, the decoding rate of an error pattern varies according to whether the first error bit in the sliding window resides at an odd or even offset in the original frame (not the offset in the sliding window).

This is illustrated in Figure 4.6, which shows the delivery rates of three error patterns as we move the error patterns to start at varying locations from 30 to 40. As shown in Figure 4.6(a), the decoding rate of an error pattern with 5 erroneous bits represented by the black line in the figure is 0 if it starts from an odd offset and 0.85 if it starts from an even offset. When using 2/3 FEC, the decoding rate of an error pattern varies depending on the offset modular 3, as shown in Figure 4.6(b). For instance, the decoding rate of an error pattern represented by the red curve, which has 4 bits in error, is 0.75, 0, and 0.55, respectively, as the offset modular 3 varies from 0 to 2. Similarly, the decoding rate varies with the offset modular 4 for 3/4 FEC, and varies with the offset modular 6 for 5/6 FEC. The cycles in the delivery rate patterns correspond exactly to the number of bits in the puncturing pattern for each convolutional code (*e.g.*, 1/2 FEC has a puncturing pattern of [1 1], 2/3 FEC has a pattern of [1 1 1 0], 3/4 FEC has a pattern of [1 1 1 0 0 1], and 5/6 FEC has a pattern of [1 1 1 0 0 1 1 0 0 1]) [1].

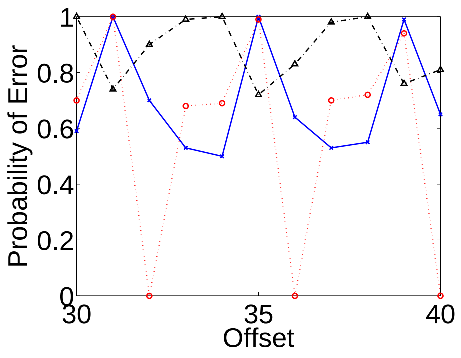
Therefore, we maintain 2 lookup tables for 1/2 FEC, 3 lookup tables for 2/3 FEC, 4 lookup tables for 3/4 FEC, and 6 lookup tables for 5/6 FEC. For each sliding window, we find the first error in the window, and identify its offset in the original frame to select the appropriate table to look up.



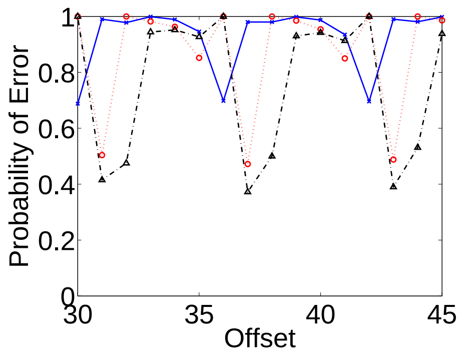
(a) 1/2 FEC



(b) 2/3 FEC



(c) 3/4 FEC



(d) 5/6 FEC

Figure 4.6: The decoding success depends on not only the error pattern in a sliding window, but also where the error pattern starts in a frame. Different curves correspond to different error patterns in a sliding window. They are shifted to different offsets in a frame.

Supporting a large sliding window: Using the above techniques, we can reduce the lookup tables for 2/3, 3/4, and 5/6 FEC to reasonable sizes. The sliding window for 1/2 FEC has 75 bits, and the number of errors in the middle cases ranges from 5 to 10. The corresponding lookup table sizes are still too large. To further reduce the table build-up time and storage cost, instead of using a sliding window of 75, we split the window into two parts: 40 bits and 35 bits. We enumerate error patterns in the first 40 bits, but only consider the number of errors in the next 35 bits (as opposed to the exact error patterns) to reduce the cost.

This yields a 2-dimension lookup table, where the first dimension is the error pattern in the first 40 bits and the second dimension is the number of errors in the next 35 bits. In our implementation, the first window has error patterns involving 4-7 errors, whereas the second window has up to 6 errors. To further reduce the overhead, we only consider the second window when the first window has 4 or 5 errors. When the first window has 6 or 7 errors, the delivery rate is already low and the impact of the second window is small. In this way, the total lookup tables for all FEC schemes is around 58 MB before compression and 8 MB after compression, which is quite affordable (*e.g.*, iPhone 5 and 6 have 1 GB RAM).

4.2.1.5 Estimating frame delivery rate

How to compute the delivery rate for a frame, which spans multiple sliding windows? One approach is to approximate it using a product of delivery rates over all sliding windows because all the sliding windows should succeed in order for a frame to succeed. However, since the delivery rates over adjacent sliding windows are not independent, this approximation is inaccurate.

To improve the accuracy, we avoid counting the same error pattern multiple times as the window slides. More specifically, we move the starting position of the sliding window to the first error and look up the delivery rate based on the error sequence within this window. Next, we move the sliding window to start from the next error. If any new error appears in the window, we look up the probability of error for the new pattern. If no new errors appear, we ignore the current sliding window and move the next sliding window to start from the next error and repeat the process until we have gone over all the errors. The product of delivery rates across all such windows gives us the final frame delivery rate estimate.

4.2.2 Method 2: Machine Learning Based Estimation

Our second method uses machine learning to estimate delivery rate based on the CSI. The advantage of this method is that the online computation is fast once we learn the function offline. Our main tasks involve selecting appropriate features and machine learning algorithm. A natural choice of features are SNR across all subcarriers. However, this requires us to learn a new delivery rate function for each modulation and FEC. Moreover, mapping from SNR to BER is quite involved, as shown in Section 4.2.1.3, and not easy to learn. Since BER has more direct relationship with the frame delivery rate, we leverage our domain-knowledge of mapping SNR to BER, as described in Section 4.2.1.3, and use BER per bit as features to map to the frame delivery rate.

Different bits have different BER due to different subcarriers they are assigned to and different bit positions in the symbol. Due to the use of OFDM and the nature of frequency diversity, BER per bit repeats every OFDM symbol. So we use BER for each bit in an OFDM symbol as the features. We choose neural network as the machine learning algorithm. The advantage of using neural networks is that they can approximate any function with arbitrary accuracy [19, 44], which is appropriate as the frame delivery rate functions are non-linear.

We use a feedforward artificial neural network model called multilayer perceptron (MLP) that maps a set of input data (BER per bit in our network) onto a set of appropriate outputs (delivery rate in our network) [11]. An MLP consists of multiple layers of nodes, where each layer is fully connected to the next layer. The first layer of nodes is called input layer and the last layer is called the output layer. The intermediate layers are called hidden layers. Each node corresponds to a neuron, with a non-linear transfer (activation) function. For our network, we use 'mapminmax' to normalize the inputs, use 'sigmoid' transfer function for hidden layers [42], and use 'purelin' as the output transfer function. The neural network is trained using Levenberg-Marquardt algorithm [31].

We use a large portion of training data to train multiple neural networks, each with different parameters, and then use the remaining portion of the training data to compute the estimation error and select the neural network that yields lowest error. This is possible since we know the ground truth of all training data.

The remaining issue is how to generate the data for training. We use both traces collected from real WiFi networks and synthetic traces generated using IEEE 802.11n TGn channel model [26]. IEEE 802.11n channel models has six profiles, including flat channel, indoor residential, residential/office, typical office, large indoor and outdoor open spaces. We use all profiles to generate channels to capture a wide range of wireless link conditions. These traces generate SNRs across subcarriers, which we then use frame-level simulation to determine the frame delivery rate. To further speed up training data generation, we also use lookup table based approach in Section 4.2.1 to generate part of the training data, and find it is much faster than frame-level simulation with only a slight increase in the error.

4.2.3 MIMO Extension

Both our estimation schemes can be applied to support MIMO as follow. To support MIMO diversity, we use the CSI values to derive the post-processed SNR (pp-SNR) values for each subcarrier under each supported transmission configuration to capture the effect of MIMO processing. In MIMO, since a transmitted symbol is received on multiple antennas, the final SNR experienced by the symbol is the combination of the multiple receptions and the combined SNR dictates whether it will be decoded correctly. We apply standard formulas to compute pp-SNR in MIMO based on the original CSI [55]. Then we derive BER according to pp-SNR as in Section 4.2.1.3, and apply the methods described in Section 4.2.1 and 4.2.2 to calculate the final frame delivery rate.

In MIMO multiplexing, a sender stripes a large frame across multiple antennas and transmits these multiple streams simultaneously. Different streams experience different channel loss depending on their transmitter and receiver pairs. Therefore, to support MIMO multiplexing, we determine to which antenna and subcarrier each symbol is assigned based on a given interleaver (*e.g.*, WiFi interleaver) and calculate pp-SNR according to MIMO multiplexing. The remaining process, including using pp-SNR to derive BER and using BER to compute frame delivery rate, is the same as above.

4.3 Our New Interleaver

Next we propose a new interleaver that is CSI-aware. The receiver feeds back the CSI according to the IEEE 802.11n standard. Instead of blindly interleaving the bits regardless of the channel condition, the sender uses our new interleaver to spread erroneous bits as far apart as possible to reduce bursty errors and improve decoding rate. The interleaving pattern does not need to be transmitted, since the receiver can derive it based on the most recent CSI it fed back and was acknowledged. Since the interleaving computation is very fast, the interleaving can be re-computed upon every CSI update.

Our interleaver sorts the bits in a decreasing order of BER. The bits are spread onto the OFDM subcarriers to maximize the distance between the high error bits as follows:

- Following the standard interleaver, we create a table of size $r \times c$, where r is the number of bits per symbol based on the modulation and c is the number of subcarriers.

- But different from the standard interleaver, we sort the bits and re-arrange the sorted bits in column-wise from top to bottom in the table. For example, the highest BER is at (1,1), the second highest BER is at (2,1), and so on.

- Next we re-arrange the columns to maximize the distance between the columns with high BER. We keep the first column at its initial location, move the second column to the center column so that the distance between the two highest BER columns are separated by $c/2$. Note that moving the second highest error column to the last column is not good since it will be close to the highest error bit in the next OFDM symbol. Then we use the first, center, and last columns as anchors, and calculate the middle columns between them and place the next two worst columns at these column indices (*e.g.*, the third worst column is moved to $round(c/4)$-th column, and the fourth worst column is moved to $round(3c/4)$-th column). This process continues recursively until all locations have been filled. In this way, the columns with larger BERs have larger separation between them. One caveat in this step is that since we observe the delivery ratio depends on the exact offset as shown in Figure 4.6, we shift the columns around to avoid occupying bad offsets. For example, for 1/2 FEC, errors at even offsets yield much higher decoding failures than errors at odd offsets.

So whenever the calculated offset is even, we shift it up or down by 1 column as long as that column has not been occupied. If the nearby columns have been both occupied, we just place it at the originally computed column. Since we place columns in a decreasing order of BER, we minimize the chance of placing the worst few bits at even offsets.

- Finally, we read the table row-wise and map the bits to the subcarriers sequentially, where each subcarrier is assigned the number of bits the current modulation can support. This step is similar to the standard WiFi interleaver to reduce bursty errors.

4.4 Rate Adaptation

In this section, we develop rate adaptation that takes advantage of our enhanced delivery estimation and interleaver. Our goal is to select the data rate for the next transmission in order to maximize throughput or minimize the energy consumption. To achieve this goal, the receiver first extracts Channel State Information (CSI) from the preamble of the previous frame. For each rate, it computes delivery ratio as described in Section 4.2 and derives throughput. It then selects the rate that maximizes the total throughput (*i.e.*, maximizing the product of maximum capacity and the delivery ratio) and feeds it back to the sender to use for the next transmission. To reduce computation cost, we may search the data rates close to the current rate instead of all possible rates.

Our rate adaptation works with both the standard WiFi interleaver and our enhanced interleaver. When our interleaver is used, we use CSI to derive an appropriate interleaving as described in Section 4.3, based on which we derive BER after de-interleaving, estimate frame delivery rate, and select the rate that maximizes throughput.

Data rates not only affect the throughput, but also impact energy consumption by changing the transmission or reception time. When energy is used as an objective, we can compute energy based on our delivery model and our energy model. Specifically, we first use the approaches in Section 4.2 to compute the delivery rate and estimate expected transmission time (ETT), which denotes how long it takes to successfully deliver a frame on average. Then we plug ETT into the energy model from Section 3.1.2 to obtain the transmission or receiving energy under a given data rate. Then we select the data rate that results in the minimum energy.

4.5 Performance Evaluation

4.5.1 Evaluation Methodology

In this section, we use trace driven simulation for our evaluation, where the traces are introduced in Section 4.1. We first compare the accuracy of our delivery rate estimation with effective SNR. Then we compare our interleaver with the WiFi interleaver. Finally, we compare the throughput and energy of our rate adaptation with effective SNR when applied to either the WiFi interleaver or our interleaver.

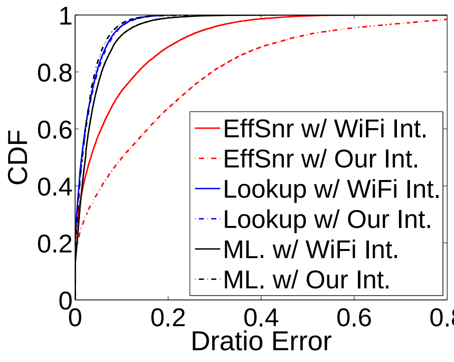
4.5.2 Performance Results

Accuracy of delivery rate estimation: We first evaluate the accuracy of our delivery ratio estimation by comparing with the ground truth. The delivery ratio is estimated for a range of CSI from the collected traces. As in the previous works (*e.g.*, [9]), we uniformly scale CSI across all subcarriers to maintain the same frequency diversity. We choose different factors for each MCS (Modulation and FEC) in order to cover the complete range of delivery ratio values from 0 to 100%. The ground truth delivery ratio is calculated by transmitting and decoding 100 frames for each scaled CSI. The ground truth is calculated for both the WiFi interleaver and our interleaver.

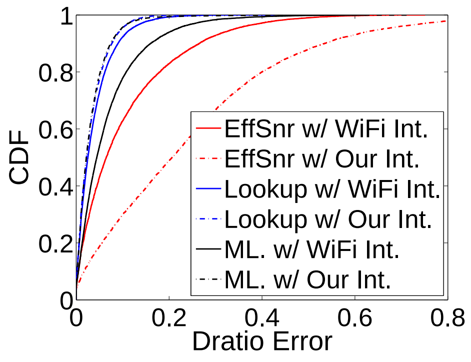
Figure 4.7(a) plots the CDF of delivery ratio estimation errors in all traces, including 27 static and 27 mobile traces with different scaling factors across all data rates using 20 MHz channel. Figure 4.7(b) summarizes the results for 40 MHz channel. Both figures compare the delivery ratio estimation errors of (i) effective SNR using the WiFi and our interleavers, (ii) lookup table based estimation using the WiFi and our interleavers, and (iii) machine-learning (ML) based estimation using the WiFi and our interleavers. The training data for 20 MHz channels come from the ground truth, and those for 40 MHz channels come from the lookup table for faster training data generation.

We make several observations. First, our two delivery ratio estimation schemes are both accurate for the WiFi and our interleaver. In 20 MHz channel, the average error of both our schemes is around 3%. The lookup scheme exceeds 10% error in only 4% of time, and the ML scheme exceeds 10% in only 7% of time. For 40 MHz channels, the lookup scheme has an average error of 4% for both interleavers, and the ML scheme has average errors of 7% and 4% for WiFi and our interleavers, respectively. The lookup scheme exceeds 10% error in 7% and 5% of the cases for WiFi and our interleavers, respectively. The ML scheme exceeds 10% error in 22% and 5% for WiFi and our interleavers, respectively. The average error in ML is 3% higher than the lookup scheme in 40 MHz channels using the WiFi interleaver in part due to the error in the training data generated from the lookup tables (which can be reduced by using frame-level simulation to generate training data). Its error is almost the same as the lookup scheme in the other cases. Due to its fast speed and similar estimation error, ML is a preferred scheme for practical use.

In comparison, the estimation error of effective SNR is much higher: its average errors are 8% and 17% for the WiFi and our interleavers, respectively. For 20 MHz channels, its estimation error exceeds 10% for 27% of the cases for the WiFi interleaver and 50% for our interleaver. For 40 MHz channels, its average errors increase to 10% and 25% for WiFi and our interleavers, respectively. The number of cases where its error exceeds 10% is 37% for WiFi, and 70% for our interleaver. There is much higher variability in effective SNR. The error varies significantly across different channel widths, across different interleavers, and even across different runs with the same settings. The effective SNR has larger estimation error with our interleaver because it is tailored to the WiFi interleaver and cannot take advantage of an enhanced interleaver, whereas our estimation schemes can incorporate any interleaver as an input to achieve high accuracy.



(a) 20 MHz

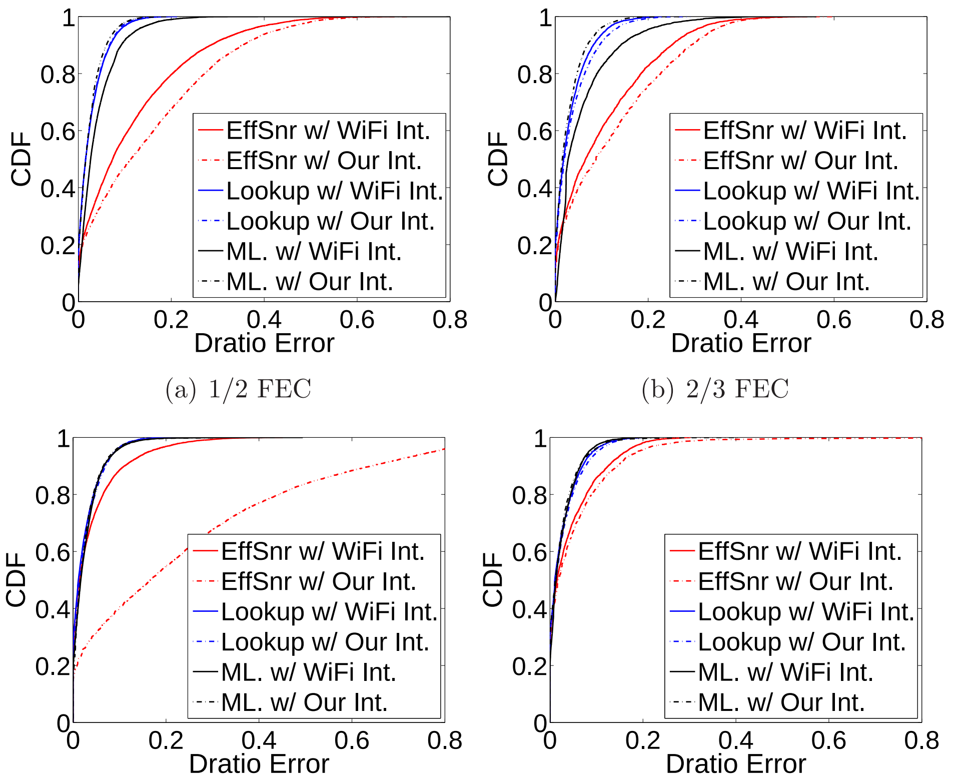


(b) 40 MHz

Figure 4.7: Comparison of delivery rate estimation across all FEC.

Figure 4.8 compares the accuracy for each FEC separately in 20Mhz. For each FEC, we consider only the modulations supported by IEEE 802.11n for that FEC. Our two schemes have similar delivery ratio estimation error for both the WiFi and our interleavers across all FEC. Their average errors are between 2%–6%, and the percentage of cases exceeding 10% error are between 2%–17%.

In contrast, effective SNR has significant variation across FEC and interleavers. For example, in 1/2 FEC, effective SNR yields an average error of 11% and 42% of the time has error exceed 10% for the WiFi interleaver. For our interleaver, the average error increases to 15%, and 53% of time has error exceed 10%. For 3/4 FEC, the average errors of effective SNR become 4% and 28% for the WiFi and our interleaver, respectively; and the percentage of cases over 10% error are 11% and 60% for the WiFi and our interleavers,



(a) 1/2 FEC

(b) 2/3 FEC

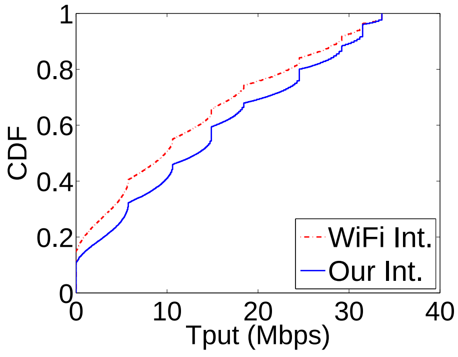
(c) 3/4 FEC

(d) 5/6 FEC

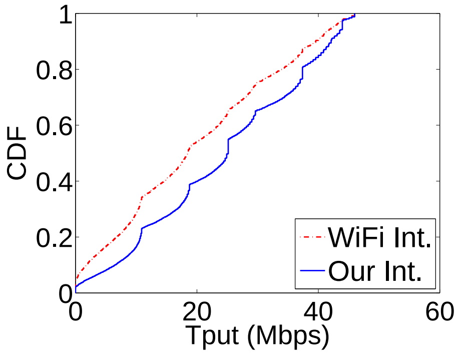
Figure 4.8: Comparison of delivery rate estimation under different FEC.

respectively. The average errors for 5/6 FEC reduces to 4% and 5% for the WiFi and our interleavers, respectively, and 14% and 18% of cases have errors exceed 10% for the WiFi and our interleavers. Effective SNR sees lowest error in 5/6 FEC because it can only tolerate very few errors and error patterns are less important in this case.

In general, large variable errors in effective SNR introduce significant uncertainty for wireless network optimization. In comparison, our estimators are more stable across all scenarios. It can significantly ease network optimization and management, and improve network predictability.



(a) 20 MHz

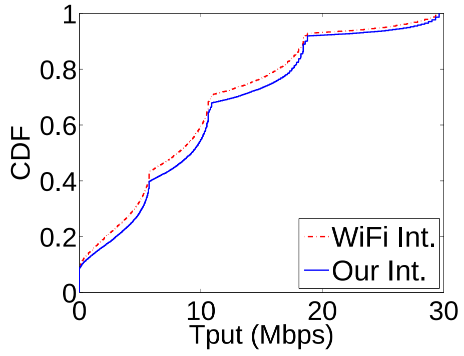


(b) 40 MHz

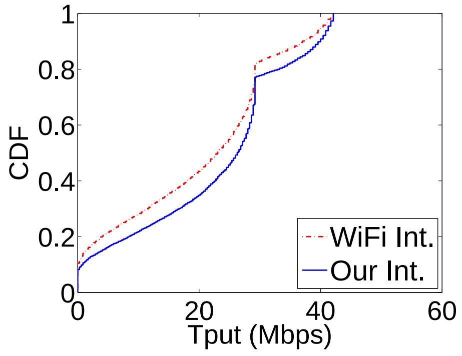
Figure 4.9: Comparison of interleavers under all rates.

Comparison of interleavers: Figure 4.9 compares the performance of our interleaver with the WiFi interleaver using all the traces across all data rates. The throughput is obtained for each interleaver by transmitting and decoding 100 frames for each CSI. The throughput is calculated for a wide range of CSI values as before. Overall, our interleaver increases the throughput over the WiFi interleaver by 18% in 20 MHz channels, and by 23% in 40 MHz channels. We expect the improvement will increase further as wider channels are used (*e.g.*, in IEEE 802.11 ac).

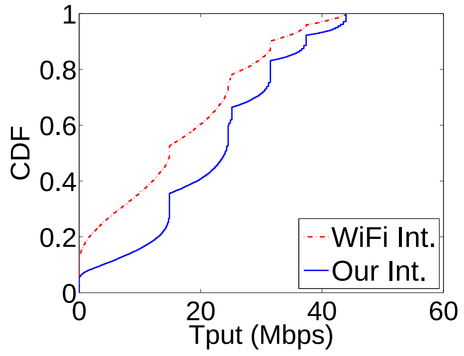
Figure 4.10 shows the interleaver performance for each FEC separately. Our interleaver out-performs the WiFi interleaver by 8.5%, 13%, 37% and 3.5% for 1/2, 2/3, 3/4 and 5/6 FEC, respectively. We see the highest improvement in 3/4 because our interleaver does a good job in spreading errors apart. Since 1/2 FEC is able to tolerate more errors, it is more robust to bursty errors. At the other end, 5/6 FEC incurs decoding failures even when there are only two errors in a sliding window, which gives less opportunities for our interleaving to optimize. In comparison, the error patterns matter more for 3/4 FEC, so it



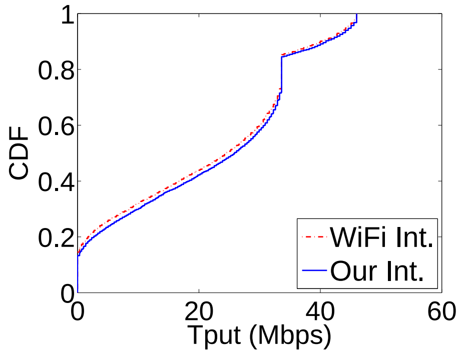
(a) 1/2 FEC



(b) 2/3 FEC



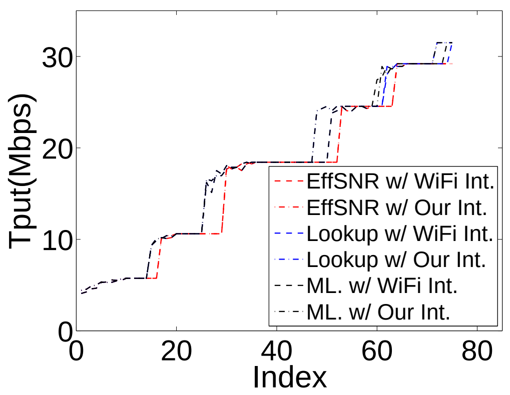
(c) 3/4 FEC



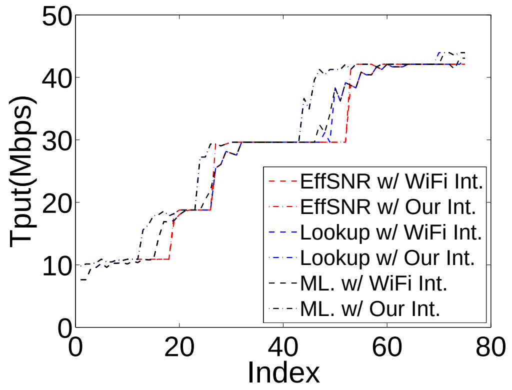
(d) 5/6 FEC

Figure 4.10: Comparison of interleavers under different FEC.

benefits most from spreading the most erroneous bits apart by our interleaver.



(a) 20 MHz

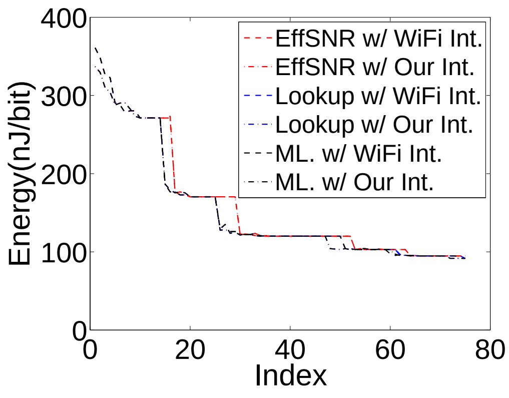


(b) 40 MHz

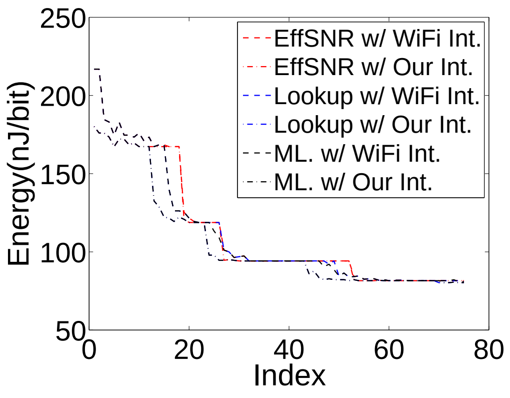
Figure 4.11: Throughput comparison of rate adaptation schemes.

Comparison of rate adaptation schemes: Finally, we evaluate the benefit of our rate adaptation that leverages our delivery rate estimation and interleaver. For ease of illustration, we select two links: a 20 MHz link and a 40 MHz link. The other links exhibit similar performance, and their results are omitted in the interest of brevity. Figure 4.11 shows the rate curves of each scheme for these links. We obtain the rate curves as follow. We impose different scaling factors for each trace and obtain throughput from different schemes. We then plot the sorted throughput for each scheme in the figure. In all cases, the rate for the next transmission is selected using the CSI of the previously received frame. Our scheme shows significant throughput improvement around the rate transition regions (*i.e.*, the boundary between two rates): the improvement ranges between 5–40%. During the non transition regions (*i.e.*, the regions that are far from the boundaries), the improvement is much smaller since the available rates are coarse-grained: two adjacent rates differ by at least 3 dB and it is hard to get close to 3 dB gain just by improving delivery rate estimation and interleaving. The benefit of our scheme will increase with more available rates. In addition, comparing the results from 20 MHz with those from 40 MHz, we observe larger benefit of our rate adaptation in 40 MHz channels since our interleaver can spread high error bits farther apart.

Finally, we compare the energy consumption of rate adaptation. The data rate affects energy by changing the time it takes to transmit or receive a frame. The longer it takes, the more energy a sender/receiver will consume. We assume the access point (AP) will run the rate adaptation scheme, regard-

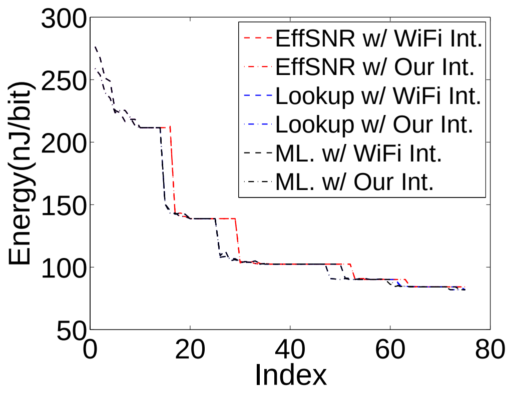


(a) 20MHz

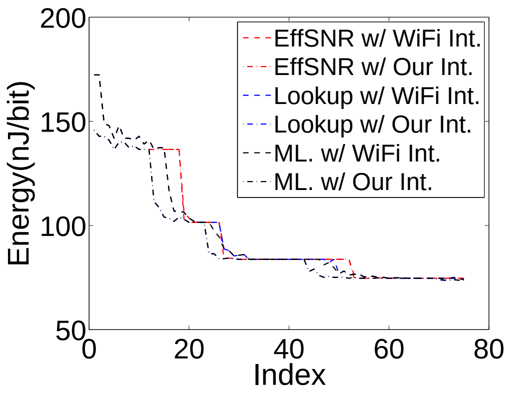


(b) 40MHz

Figure 4.12: Transmission energy comparison of rate adaptation schemes.



(a) 20MHz



(b) 40MHz

Figure 4.13: Receiving energy comparison of rate adaptation schemes.

less if the AP is transmitting or receiving. That is, if the AP is transmitting, the AP selects the rate to reduce transmission time so that the receiving client can save receiving energy; if the AP is receiving, the AP selects the rate and feeds the selected rate back to the transmitting client to use for the next transmission to save the client's transmission energy.

We use the energy model in Section 3.1.2 to derive the client's energy consumption based on expected transmission time (ETT) under different rate adaptation schemes.

Figure 4.12 and 4.13 compare transmission and receiving energy of different schemes, respectively. As we would expect, similar to the throughput results, the energy improvement takes place around the rate transition regions, where our rate adaptation reduces transmission energy by 7%–33% and reduces receiving energy by 6%–32% over the effective SNR with the WiFi interleaver.

Chapter 5

Rate Adaptation with Partial Retransmission

5.1 Retransmission Characterization

5.1.1 Motivation

Wireless channel condition is hard to predict due to mobility, dynamic interference, and environmental changes. Therefore, transmission failures are common in wireless networks. Traditional systems retransmit an entire frame upon every failure. This is inefficient since a significant portion of bits may have already been correctly received and retransmitting these bits is wasteful. Recognizing this inefficiency, PPR [50] proposes to extract physical layer hints to determine if bits are correct and discards/retransmits the bits with lower confidence. PPR is an interesting concept. Maximizing its full potential requires addressing several important challenges.

- It is hard to tell exactly which bits are corrupted. In order to ensure all incorrect bits are retransmitted, a conservative threshold has to be used.

[0]This chapter revises the following previously published material. Smart Retransmission and Rate Adaptation in WiFi. Muhammad O. Khan, Lili Qiu, Apurv Bhartia, Kate Lin. IEEE ICNP, San Francisco, USA, November 2015. I was involved in the design and implementation of the proposed rate adaptation and retransmission protocol. Apurv Bhartia assisted in the evaluation of the proposed scheme. Kate Lin helped in project-related discussions. Lili Qiu, supervised the entire project.

Many bits below the threshold may well be received correctly. Completely discarding them is wasteful. It would be beneficial if we can take a step further by leveraging information from bits with lower confidence to further reduce retransmission overhead. In other words, we should not only support partial retransmission, but also combine bits from multiple failed transmissions.

- We observe that a receiver decodes data by first demodulating the incoming signals to bits and then performing FEC decoding, as shown in Figure 5.1. Combining can be applied to the raw analog signal, the demodulated data before FEC, or the FEC decoded data. Where should combining take place to maximize the gain while achieving high flexibility?

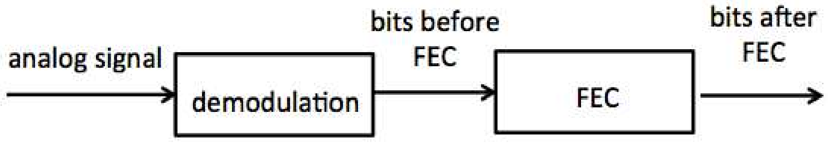


Figure 5.1: Process of data decoding.

- How do partial retransmission and combining affect rate adaptation? Existing schemes (*e.g.*, PPR [50] and SOFT [98]) show significant performance benefit under a fixed data rate. However, their gains diminish under standard rate adaptation, which tends to pick a conservative rate and leaves little opportunity for partial retransmission and combining. Therefore we should re-visit rate adaptation to take advantage of partial retransmission and combining.

5.1.2 Our Approach

We seek to address these challenges in turn. We propose combining-aware partial packet recovery in which we retransmit the bits with lower confidence. For any bits that are received more than once, we combine them to improve accuracy.

To maximize the combining gain, we examine how combining accuracy changes when combining takes place at different stages. We find the combining gain is highest when it is performed before FEC decoding. However, combining analog signals has several limitations: (i) it is unclear how to combine signals transmitted using different modulations or FEC codings since it requires converting signals from one modulation/FEC coding to another before combining and there is no known solution to perform such conversion while maintaining the confidence of demodulated bits, and (ii) it is unclear how to combine signals from partial retransmissions because when different transmissions contain different signals, the same bit may be spread over different parts of the signals (*e.g.*, a bit may be the first bit in a QAM-16 symbol during the first transmission, but the second bit in a QAM-16 symbol during the second transmission). Bit position in a symbol is important since it affects the confidence level.

Therefore we propose combining the demodulated bits before FEC so that it is independent of modulation, thereby simultaneously achieving high combining gain and flexibility. In particular, we estimate Log-Likelihood Ratio (LLR) for each demodulated signal based on the signal-to-noise ratio (SNR) obtained from the preamble. To support both partial retransmission and combining, we use the LLR to determine which bit(s) before FEC are likely to be corrupted and combine bits from multiple transmissions by adding up their LLR and feeding the resulting LLR to a FEC decoder.

To maximize effectiveness, we observe that traditional rate adaptation tries to select the rate that maximizes throughput of the current transmission, where throughput is computed based on packet delivery rate. This essentially means that utility is 0 when the current transmission fails.

This usually results in a conservative rate, which leads to successful delivery of the frame in one try and leaves little opportunity for partial packet recovery and combining. In order to fully take advantage of combining, the rate adaptation design should be revisited. In particular, the use of combining means that a failed transmission still conveys useful information by increasing LLR and has positive utility. To capture this notion, we formulate a new rate adaptation problem whose goal is to minimize the total time of the transmission and retransmissions to deliver a frame. This significantly changes the rate adaptation problem. We develop a novel rate adaptation scheme to select the rates for all the transmissions associated with a frame in order to maximize throughput, or equivalently, to minimize expected transmission time. Our main insight is that by leveraging combining, we can potentially use a more aggressive data rate due to a much lower retransmission cost. This creates an opportunity for partial retransmissions and combining.

5.2 Signal Combining

Signal combining is a well-known technique in wireless communication and is used to provide spatial and temporal diversity. Signal combining leverages multiple receptions of the same data to get a better estimate of the received signal. In this section, we first describe and compare different combining approaches. Then we present our approach to combine partial retransmissions while maximizing the combining gain.

5.2.1 Different Combining Schemes

Overview of three combining schemes: As mentioned in Section 5.1.1, signal combining can be performed at various stages of decoding: (i) before demodulation, (ii) after demodulation but before FEC decoding, and (iii) after FEC decoding.

In (i), the same signal should be transmitted. The receiver computes a weighted sum of the received signals, and then demodulates the resulting signal as usual. It is commonly referred to as symbol combining. Figure 5.2 shows an example of (i), where two BPSK signals $R1$ and $R2$ are combined and demodulated correctly to 1.

In (ii), a receiver first demodulates an analog signal into digital bits. Based on the distance between the received signal and constellation point, the receiver computes log-likelihood (LLR) value. LLR is the logarithm of the ratio

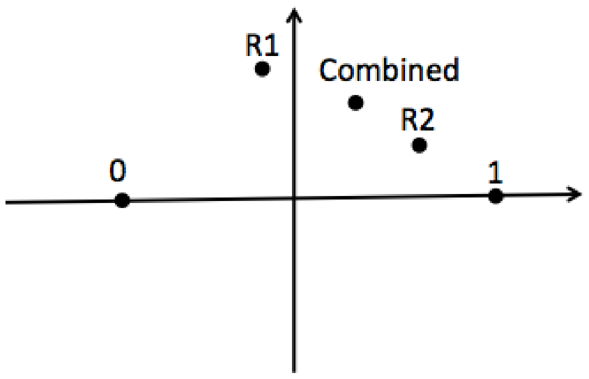


Figure 5.2: Symbol combining.

of probabilities of a 0 bit being transmitted versus a 1 bit being transmitted given a symbol r is received. The LLR for a bit b is computed as follows:

$$L(b) = \log\left(\frac{Pr(b=0|r=(x,y))}{Pr(b=1|r=(x,y))}\right) \tag{5.2.1}$$

where r is the received signal, (x,y) is the x and y coordinates of the received signal in the constellation map, b is the transmitted bit. If we assume the probabilities of all symbols are equal, the LLR of an AWGN channel is:

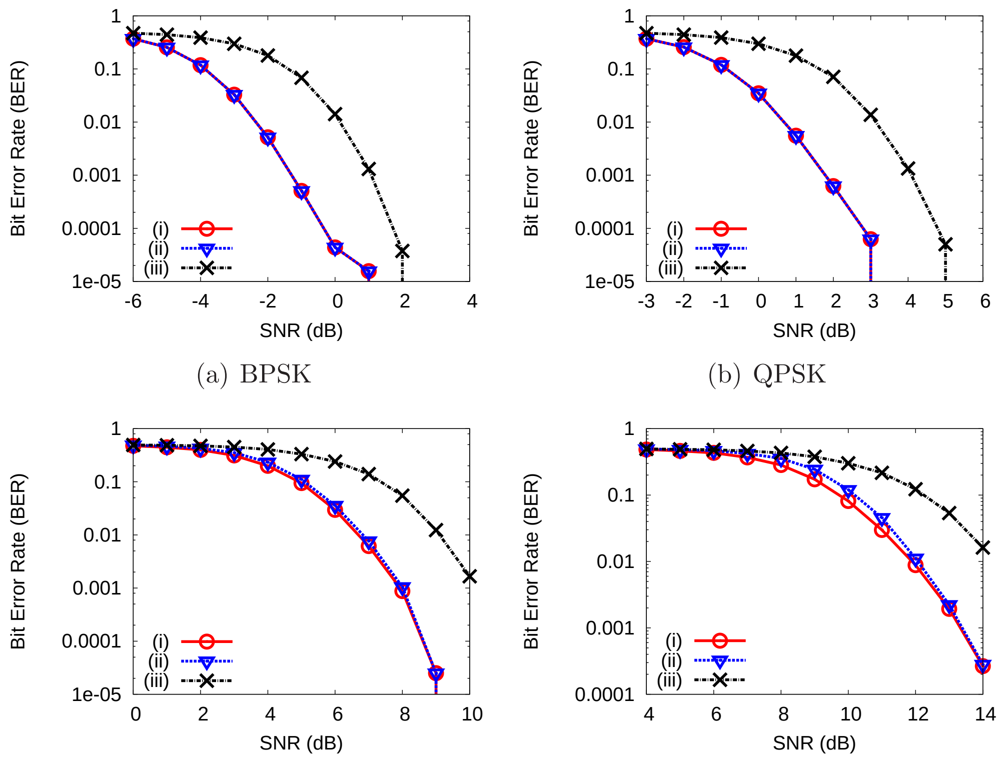
$$L(b) = \log\left(\frac{\sum_{s\in S_0(b)} e^{-\frac{1}{\sigma^2}((x-s_x)^2+(y-s_y)^2)}}{\sum_{s\in S_1(b)} e^{-\frac{1}{\sigma^2}((x-s_x)^2+(y-s_y)^2)}}\right) \tag{5.2.2}$$

where $S_0(b)$ and $S_1(b)$ are ideal symbols with bits 0 and 1 at the given bit position b, respectively, and σ^2 is noise variance of baseband signal and can be obtained using the preamble. To combine multiple received data, the receiver sums up LLR values from all the receptions and feeds the resulting LLR to an FEC decoder (*e.g.*, Viterbi decoder) to enhance the decoding rate. There are two types of decoders: hard decoder, which takes a hard decision as an input (*e.g.*, 0 or 1), and a soft decoder, which leverages more fine-grained LLR values to improve decoding accuracy.

Our scheme supports both hard decoder and soft decoder. We focus on the hard decoder in our evaluation since its delivery rate can be estimated accurately (which is required for rate adaptation) whereas our experiments show that the delivery rate of a soft decoder (with or without combining) depends on not only effective SNR but also the mapping between symbols and SNR and is much harder to estimate. Therefore we defer our evaluation of the soft decoder to our future work.

In (iii), a receiver performs FEC decoding on each received data separately and outputs LLR. Our evaluation uses Bahl Cocke Jelinek Raviv (BCJR) Maximum a Posteriori (MAP) decoding algorithm [5, 6]. Its performance is the same as Viterbi decoder when both use soft decoding, but slightly worse than Viterbi when both use hard decoding. Then the receiver adds up the LLR for each bit, and uses a thresholding to determine the final value for the bit (*e.g.*, the bit is 1 if the LLR sum is greater than 0, and 0 otherwise).

Comparison of the three combining schemes: We compare the combining gains using different schemes as follow. We first generate signals according to a selected modulation. We then add AWGN noise to the signal whose mag-



(a) BPSK (b) QPSK

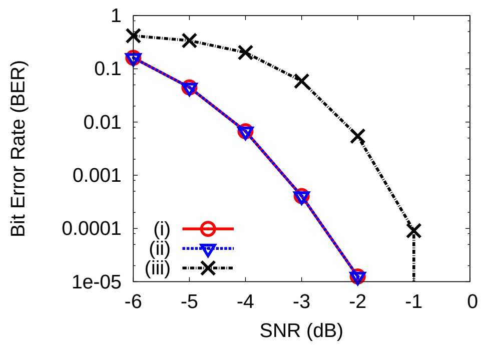
(c) QAM16 (d) QAM64

Figure 5.3: Hard combining gain comparison, where (i) combining before demodulation, (ii) combining after demodulation but before FEC decoding, and (iii) combing after FEC decoding.

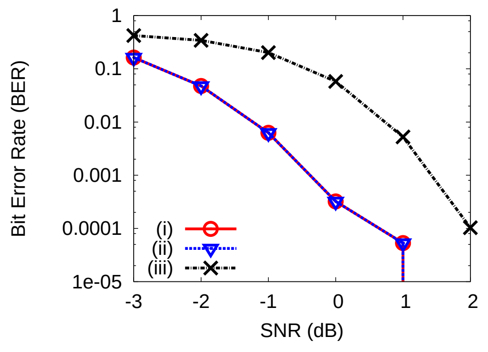
nitude is determined by SNR. We send each signal twice and then calculate the bit error rate (BER) of each scheme over 320,000 bits.

Figure 5.3 compares different combining schemes using a hard decoder, which uses 1/2 convolutional FEC code with BPSK, QPSK, QAM-16, and QAM-64. Figure 5.4 summarizes the performance of different combining schemes using a soft decoder.

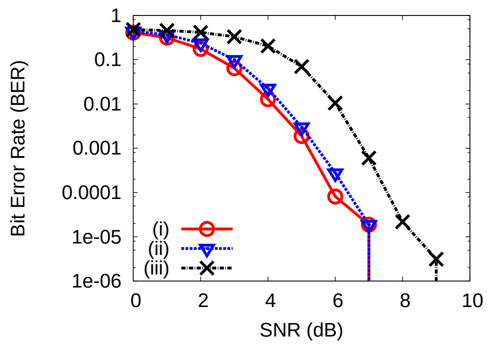
In all cases, we observe (i) $\approx$ (ii) $>$ (iii). (i) and (ii) perform similarly,



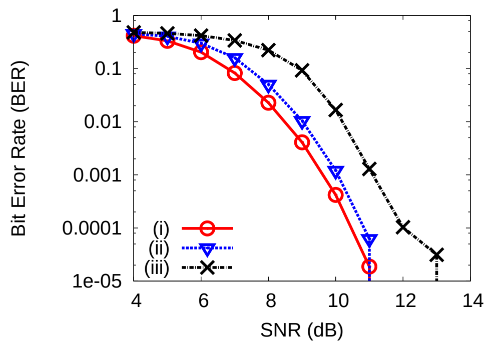
(a) BPSK



(b) QPSK



(c) QAM16



(d) QAM64

Figure 5.4: Soft combining gain comparison, where (i) combining before demodulation, (ii) combining after demodulation but before FEC decoding, and (iii) combing after FEC decoding.

and only occasionally (i) is slightly better than (ii). However, (i) is limited by the constraint that the retransmitted symbols must be the same as the original symbols. This implies that same set of symbols must be retransmitted and the retransmissions should also use the same data rate (*i.e.*, the same modulation and FEC code) as the original transmissions. This severely limits the applicability of symbol combining and makes it hard to support partial retransmission and rate adaptation for retransmissions.

On the other end, (iii) is completely independent of the data rate (*i.e.*, modulation and FEC). Data transmitted using different rates can be easily combined at the digital level. Moreover, it can easily support partial retransmissions. However, as shown in Figure 5.3 and 5.4, its combining gain is lower than (i) and (ii) by up to 2 dB since it cannot leverage the FEC coding gain.

(ii) achieves the best of both worlds. Its combining gain is close to the maximum combining gain: (ii) yields the same SNR gains for BPSK and QPSK as (i), and is at most 1 dB lower than (i) under a higher order modulation.

Furthermore, (ii) is flexible and can support partial retransmissions and different modulation as we will show in Section 5.2.2. Therefore we use (ii). Note that (ii) gives the same gain as (i) under BPSK and QPSK, but is slightly worse under QAM because demodulation is independent across different bits in BPSK and QPSK, which makes combining before and after demodulation equivalent in BPSK and QPSK, whereas demodulation of different bits is correlated in QAM and combining before demodulation allows a bit to benefit from the combining gain from other bits in the same symbol, thereby improving demodulation success rate.

5.2.2 Combining Partial Retransmissions

Partial data retransmission is another technique used to reduce retransmission overhead. In WiFi, it is common to have corrupted frames that have a small number of erroneous bits. Therefore, retransmitting an entire frame is wasteful. PPR [50] uses PHY-layer hints to identify the bits that are likely to be in error and only retransmits those bits. The concept of PPR is interesting. However, there are three major limitations of PPR.

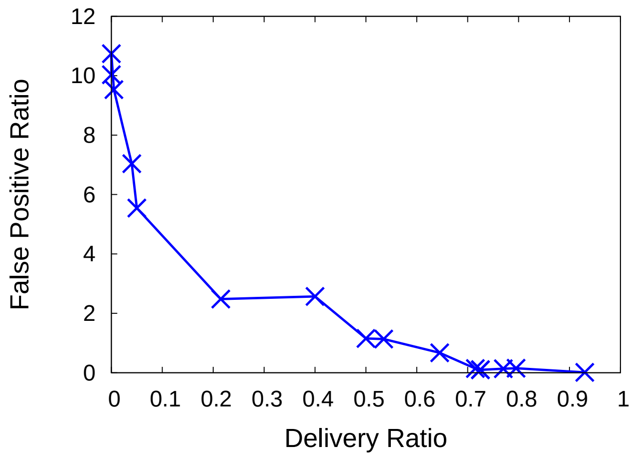


Figure 5.5: PPR false positive ratio.

First, it is hard to determine an appropriate LLR threshold (after FEC decoding) for deciding which bits are correct and which bits are in error. This is because LLR threshold must be high enough that all bits in error are chosen to be retransmitted. This is necessary because PPR selects bits for retransmission after FEC decoding. However, if a large threshold is selected, a significant number of bits that were received correctly also end up being retransmitted. Figure 5.5 plots the false positive ratio (*i.e.*, the ratio between the number of bits that are correct but deemed in error versus the total number of actual erroneous bits) versus the frame delivery rate. Each point in the curve is generated by simulating 200 frame transmissions over a given SNR. For each frame, we determine the LLR threshold (after FEC decoding) for which all erroneous bits are selected for retransmission. For the selected threshold, we determine the number of false positives and calculate the false positive ratio for that frame. Finally, we take the average of false positive ratios over all 200 frames and plot it against the delivery ratio of the 200 frames. We simulate over a range of SNR values to generate the curve.

As we can see, the false positive rate can be very high. When the delivery rate is 0.4 or lower, the false positive ratio is over 2, indicating that we may retransmit over twice the number of erroneous bits. When the delivery rate reduces further, the false positive ratio becomes 6 or higher.

Second, PPR discards the bits below the threshold. Given a significant number of bits below the threshold may well be correct, it is important to use these bits to improve decoding.

Third, PPR is applied after FEC decoding. As shown in Figure 5.3, the benefit of combining after FEC is lower than the other alternatives.

Most importantly, PPR does not modify the rate adaptation and uses the rate adaptation that is not aware of partial packet recovery. As a result, in most cases the selected rate allows a packet to be delivered successfully in one shot and gives little opportunity for partial packet recovery. Our evaluation confirms this intuition and shows PPR performs similar to WiFi under the existing rate adaptation.

Motivated by the benefit and limitations of PPR, we propose an approach to combine partial retransmissions. As PPR, we also identify likely erroneous bits and perform partial retransmissions. Our approach differs from PPR in the following ways. First, instead of using a fixed threshold to determine if a bit is correct, we search for bits for retransmissions that maximize throughput. Our approach does not require that all erroneous bits be retransmitted since combining is done before FEC in our scheme and the FEC can tolerate a few erroneous bits. We no longer need to select a threshold, but let the rate adaptation scheme automatically balances the cost of retransmission and delivery rate, as described in Section 5.3. Second, we do not discard bits likely to be received in error. Instead, we combine all the received bits to increase the data decoding probability. Third, we propose a combining aware rate adaptation scheme, which explicitly takes into account partial retransmission and combining to select a more appropriate rate.

To reduce feedback overhead, we retransmit in a unit of OFDM subcarriers. Due to temporal stability within a subcarrier and frequency diversity across subcarriers, symbols transmitted on the same subcarrier are likely to experience similar SNR and symbols transmitted on different subcarriers tend to experience different SNR. We compute Bit Error Rate (BER) of each subcarrier, and sort the subcarriers in an decreasing order of BER. Our rate adaptation, described in Section 5.3, finds the worst k subcarriers for retransmission (*i.e.*, all bits on these subcarriers are retransmitted). The feedback contains a bitmap of subcarriers, where 1 indicates the corresponding subcarrier requires retransmission. The transmitter concatenates all the bits required for retransmission in the order of OFDM symbol index and then an increasing order of subcarrier index, and maps them sequentially to all the subcarriers during retransmissions. Since the receiver selects which bits to retransmit, it knows the mapping used by the transmitter.

The combining is performed by summing up the LLR values for any bit that has more than one reception and feeding the resulting LLR values to the Viterbi decoder. By leveraging information received previously despite having low confidence, we can achieve high decoding rate and reduce retransmission overhead.

5.2.3 Benefits of Combining

In general, combining is beneficial for three main reasons. First, combining reduces retransmission cost and may potentially allow us to choose a higher data rate. Since wireless losses are probabilistic, sometimes frames can still be delivered successfully at a higher rate. Even when they are not successfully delivered, the retransmission overhead is smaller and does not degrade throughput significantly. Second, combining is useful under frequent fluctuations in the wireless channel (*e.g.*, arising from mobility). Under unpredictable wireless channel, losses are common. By leveraging information from previously failed transmission, combining reduces the cost of failures. Third, combining is also useful under dynamic interference due to reduced retransmission cost in case of collision.

The same combining scheme works for all the scenarios. In all cases, we use both preambles and data symbols to compute SNR for each subcarrier. For the preamble, since we already know the reference symbol, we simply use the distance between the expected and received constellation point to compute

SNR. For the data symbol, since we do not know the ground truth, we use the distance between the received and the few most closest constellation points to derive SNR. Then we compute the variance $\sigma^2 = 1/SNR$, and derive LLR as:

$$L(b) = \log \left(\frac{\sum_{s \in S_0(b)} e^{-\frac{1}{\sigma^2}((x-s_x)^2+(y-s_y)^2)}}{\sum_{s \in S_1(b)} e^{-\frac{1}{\sigma^2}((x-s_x)^2+(y-s_y)^2)}} \right). \qquad (5.2.3)$$

The LLR formula is general and applicable with or without interference, since the variance in the formula captures background noise when there is no interference, and captures both background noise and interference otherwise.

5.3 Combining-aware Rate Adaptation

5.3.1 Rate search

Traditional rate adaptation selects a rate to minimize the time for the current transmission. That is, it treats each transmission and retransmission as isolated units and picks a rate to optimize these individual units independently. However, we observe that multiple transmissions associated with the same frame should be considered as one unit and our goal is not to optimize individual transmissions, but to optimize the entire process of successfully delivering one frame.

The transmission time depends on the data rate used and the amount of data to be transmitted. For example, one could either retransmit more data bits at a higher rate (less reliably) or retransmit fewer bits but at a lower rate (more reliably). Therefore our rate adaptation should search not only for the data rate used for each transmission but also how much data to retransmit every time.

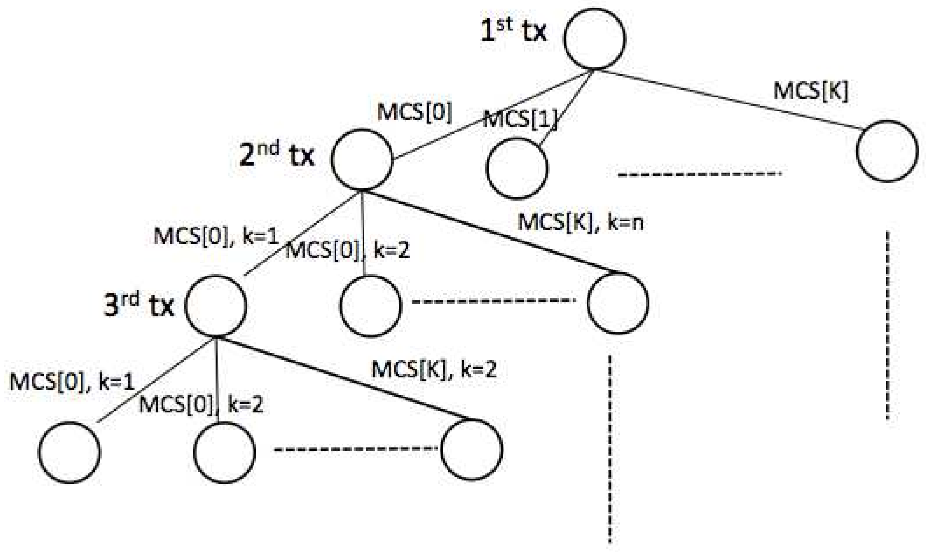


Figure 5.6: Rate search tree.

To make the search scalable and reduce feedback overhead, we retransmit in a unit of subcarriers. It means that if a subcarrier is selected for retransmission, then all the bits that are transmitted on that subcarrier during the original transmission are retransmitted. To select the subcarriers for retransmission, we calculate the average BER for each subcarrier, and then sort the subcarriers in a decreasing order of BER. We incrementally add one subcarrier at a time to retransmit, starting with the worst subcarrier (highest BER). Each time we search over all the data rates and compute the expected transmission time and delivery rate. If the delivery rate is below a threshold, we repeat the search process for the next retransmission. This process continues until we populate the rate search tree. Figure 5.6 shows an example of rate search tree, where each branch of the tree corresponds to a transmission strategy. As it shows, the left-most branch corresponds to transmitting at the rate of MCS[0] for the first time, transmitting the worst subcarrier in the first transmission at the rate of MCS[0] for the second time, and transmitting the worst subcarrier (after combining) at the rate of MCS[0] for the third time. The branch terminates as soon as all the bits in a frame are successfully delivered or the maximum number of retransmissions is reached. Our evaluation considers frame delivery rate over 99% or 2 retries as the stopping criterion. Once we build the rate search tree, the best transmission strategy for a frame is simply the branch that yields the minimum transmission time. It should be noted that the BER based ordering of the subcarriers may change after retransmission, as the receiver combines all the transmissions associated with the same bit and re-sorts the subcarriers according to the average BER (after combining).

The transmission time of each branch is calculated as:

$$TotalTime = \sum_i Time_i = \sum_i (size_i/rate_i + overhead_i) \tag{5.3.1}$$

where $size_i$ and $rate_i$ are size and data rate of the i-th transmission and $overhead_i$ is the overhead of the i-th transmission. $size_i$ depends on the delivery rate after combining all transmissions that have occurred so far. $overhead_i$ includes the header overhead and MAC overhead such as DIFS,

modulation	BER
BPSK	$Q(\sqrt{2snr})$
QPSK	$Q(\sqrt{snr})$
QAM-16 (bits 0 and 2)	$\frac{1}{2}Q(\sqrt{snr/5})$
QAM-16 (bits 1 and 3)	$Q(\sqrt{snr/5})$
QAM-64 (bits 0 and 3)	$\frac{1}{4}Q(\sqrt{snr/21})$
QAM-64 (bits 1 and 4)	$\frac{1}{2}Q(\sqrt{snr/21})$
QAM-64 (bits 2 and 5)	$Q(\sqrt{snr/21})$

Table 5.1: BER as a function of SNR for different modulation

SIFS, and feedback overhead. Our implementation has two types of feedback: partial ACK and complete ACK. A partial ACK contains one OFDM symbol to specify the bitmap of which subcarriers to retransmit and the data rate to use for retransmission. A complete ACK is the same as a WiFi ACK.

5.3.2 Computing delivery ratio

An important step in rate search is to compute the delivery rate. We compute it as follows.

- Derive uncoded BER (*i.e.*, BER before FEC decoding): Our goal is to compute the probability of each bit being in error after combining multiple transmissions. To achieve this, we (i) derive SNR for each bit, (ii) sum up the SNR of all the transmissions involving the same bit, and (iii) map the combined SNR to BER using Table 5.1.

  Below we elaborate step (i). For the received bits, we calculate the LLR value for each bit using the expression (5.2.1). We then calculate the probability of having an error in a received bit using:

$$P_{\mathrm{b}}(k) = \frac{1}{(1 + e^{|s_k|})} \tag{5.3.2}$$

where $|s_k|$ is the absolute LLR value of bit k and $P_{\mathrm{b}}(k)$ is the probability of bit k being in error [94]. We then use the inverse of the formulas in Table 5.1 to map BER to SNR of individual bits.

Note that we calculate the BER separately for each bit location in a given symbol. For example, each QAM-16 symbol has four bits $b0$, $b1$, $b2$ and $b3$. As shown in Table 5.1, the BER experienced by $b0$ and $b2$ is different from that of $b1$ and $b3$. This is because the way constellation points are located as shown in Figure 5.7. Therefore, it is necessary to have a separate BER estimate for each bit location to achieve high accuracy. We empirically derive BER for QAM as shown in Table 5.1.

To compute the uncoded BER of future transmissions, which is required for rate search, we use the channel state information (CSI) of the previous frame to make prediction. The CSI consists of $M \times N$ matrices H_s, each of which specifies amplitude and phase between pairs of N transmitting and M receiving antennas on subcarrier s. The CSI is estimated using the preamble of a frame, and allows us to compute SNR for each subcarrier. For simplicity, we use the previous CSI for prediction. Alternatively, one may also use exponential weighted moving average (EWMA) (*i.e.*, $y = \alpha x + (1 - \alpha)y$, where x is the current sample, y is the prediction) or Holt-Winter algorithm [53] for prediction.

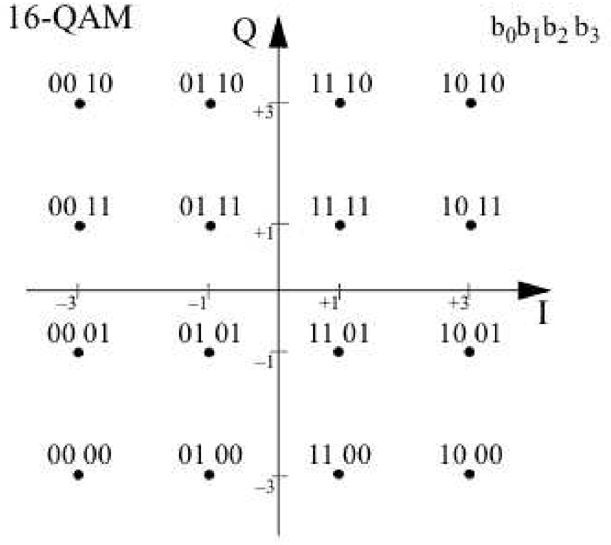


Figure 5.7: Constellation points for QAM-16.

- Compute Effective SNR: We calculate the effective SNR by averaging BER calculated in the first step and then mapping it back to the SNR for each modulation using the procedure specified in [38].
- Lookup Delivery Ratio: We use a standard pre-computed lookup table (*e.g.*, as in [38]) to get the delivery ratio for each data rate based on the effective SNRs calculated in the previous step. The table contains delivery ratios for SNR values over a range of −10 to 30 dBs and packet sizes from 25 to 4000 bytes. The table is generated offline in Matlab by sending and decoding 1000 packets for each SNR value over a flat fading channel. This step is the same as [38].

The above steps work for both SISO and MIMO. Also, the last two steps can be replaced by the delivery ratio estimation techniques we propose in Section 4.2, which will result in improved performance. The techniques in Section 4.2 were developed after these evaluations.

Speed up: To enhance the efficiency of our search, we prune the search tree in the following way. We perform breath-first search over the rate search tree and keep the top M branches at each level in terms of throughput so far (computed as the expected number of bits delivered so far divided by the transmission time including the overhead), and prune all the other branches. We only explore the top branches deeper. We control the value of M to balance the tradeoff between the computation time and optimality. Our evaluation uses $M = 4$. Furthermore, when we search for the number of subcarriers to retransmit, we stop whenever adding a subcarrier to retransmit reduces throughput. Finally, we limit the depth of a tree by looking ahead 2 transmissions.

Energy minimization: Our rate search is general and can be used to optimize different objectives. In particular, we can use the same search algorithm to minimize energy consumption. We use the energy model proposed in Section 3.1.2 to compute the energy consumption. We compute ETT for each transmission to derive energy and select the branch that leads to the lowest energy. Our evaluation in Section 5.5 shows our approach saves considerable energy.

Supporting MIMO: Our rate search can support MIMO. There are two differences in MIMO. First, there are more rates under MIMO, and the rate search tree becomes larger. So careful pruning is important to keep computation time low. For example, we can search around the neighborhoods of the previous rates. Second, we use MIMO post-processed SNR (pp-SNR) to compute the new delivery rate after performing a new retransmission under MIMO. In spatial diversity, transmissions between different transmitter and receiver antennas are spatially combined and we can apply the formulas in [55] to compute pp-SNR. In spatial multiplexing, a sender stripes a large frame across multiple antennas and transmits these multiple streams simultaneously. We derive BER for each subcarrier as usual, and compute effective SNR by averaging BER over all subcarriers and all spatial streams. As the data rate increases in MIMO, frame aggregation can be used (as usual) to minimize MAC/PHY overhead in the retransmission.

5.4 Protocol Implementation

We implement our approach in USRP [92]. To reduce feedback overhead, we let the receiver measure log-likelihood, perform rate search, and feed back the selected rate to the sender. The receiver feedback not only includes the data rate for retransmission but also which subcarriers to retransmit using a simple bitmap, where 1 at the i-th position indicates that the sender should retransmit data that was sent on the i-th subcarrier in the original transmission. To enhance reliability of feedback, we transmit the feedback at

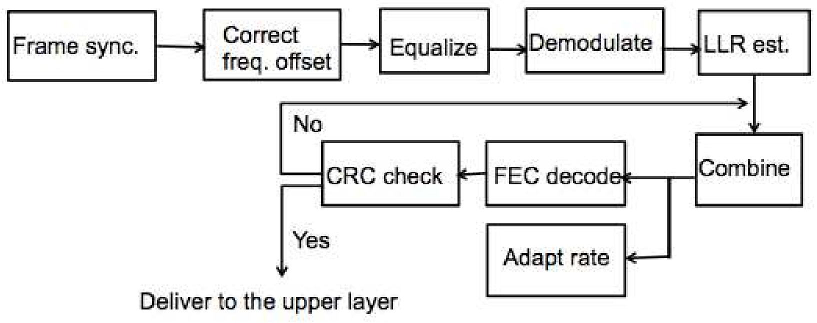


Figure 5.8: USRP implementation

the lowest data rate in our implementation.

As shown in Figure 5.8, the receiver processes the incoming signals as follow. It first uses cross correlation to detect the beginning of an incoming frame [32], corrects for frequency offset [32], estimates the channel coefficient by dividing the received preamble by the known preamble. Then it uses the same preamble to compute the noise variance, demodulates the analog signals to digital bits, and derives LLR for each bit. It further combines the LLR values for the bits that have been received earlier and passes to FEC decoder.

If it fails, the bits and their LLR values are stored for future combining. Otherwise, they are delivered to the upper layer. Meanwhile, based on the combined results and latest channel estimate, the receiver selects the rate as described in Section 5.3.

Our implementation uses a bandwidth of 1 MHz, 80 OFDM subcarriers, FFT window of size 128, cyclic prefix of 32 samples, and FPGA running at 100MHz. We use the convolutional coding implementation from the IT++ public library [49]. 2.49GHz frequency is used to avoid external interference from the campus network. We modify the default GNU Radio OFDM implementation to realize symbol combining, receiver feedback, retransmission, and rate adaptation.

5.5 Simulation Evaluation

5.5.1 Evaluation Methodology

In this section, we first use trace driven simulation to compare different schemes in Matlab. In Section 5.6, we further compare using USRP implementation. We collect three channel traces from static environments, and another three traces from mobile environments with human walking speed. The three mobile traces are collected in an office environment using 1 moving receiver and 3 static senders. The three static senders are 7m away from each other. Each trace corresponds to one of the three senders transmitting while the receiver is moved at a walking speed.

We use Intel WiFi Link 5300 (iwl5300) IEEE 802.11 a/b/g/n wireless network adapters to collect the CSI of each frame preamble across all subcarriers. These NICs have three antennas. We use 802.11n, and enable all three antennas at both the sender and receiver. The modified driver [39] reports the channel matrices for 30 subcarrier groups, which is about one group for every two subcarriers in a 20 MHz channel according to the standard [1] (*i.e.*, 4 groups have one subcarrier each, and the other 26 groups have two subcarriers each). We use a transmission power of 15 dBm. Three MIMO streams are sent, so the NICS report CSI in the form of 3×3 matrices for each frame. In our evaluation, we account for the SIFS/DIFS and feedback overhead, which includes MAC-layer ACKs and feedback of which subcarriers to retransmit.

We compare the following approaches. All schemes except Smart and SmartNoCombine use Effective SNR [38] based rate adaptation, which is state-of-the-art rate adaptation scheme and is shown to capture frequency diversity quite accurately. Effective SNR maps the SNR of each subcarrier to BER, averages the BER across subcarriers, converts it back to SNR, and uses effective SNR to look up the frame delivery rate table.

- WiFi: Whenever a frame does not pass CRC check, the entire frame is retransmitted. The receiver discards a failed transmission, and decodes a retransmission independently.
- SOFT: When a frame is corrupted, the entire frame is retransmitted. It combines symbols from all transmissions using maximal ratio combining (MRC) and then performs demodulation and FEC decoding on the combined signals. If the combined result passes CRC, it is delivered to the upper layer. Otherwise, an entire frame is retransmitted until it succeeds.
- PPR: Low-confidence bits are retransmitted. The receiver uses a confidence metric like LLR to extract the bits likely to be correct from each reception and merges them together. Our implementation differs from

the original PPR [50] in that (i) we use LLR values from demodulation to identify the low-confidence bits and retransmit coded bits (*i.e.*. before FEC decoding), which is better than after FEC decoding (done in PPR), and (ii) we retransmit in units of subcarriers to reduce feedback overhead.

- Smart retransmission and rate adaptation (Smart): This is our main approach as described in Section 5.3, which supports combining partial retransmissions before FEC decoding and combining-aware rate adaptation.

- SmartNoComb: This approach is the same as Smart, but only enables partial retransmission and disables combining. The rate adaptation also uses a similar tree-based rate search in Section 5.3 except that we do not consider the combining gain (*i.e.*, the retransmission bits are taken as they are without combining with the previous receptions). The difference between PPR and SmartNoComb reflects the benefit of partial retransmission aware rate adaptation, and the difference between SmartNoComb and Smart reflects the combining gain.

- SmartUnaware: This is the same as Smart Retransmission except that the rate adaptation uses effective SNR per transmission and does not optimize rate selection across multiple transmissions. This is a useful reference for us to quantify the benefit of combining-aware rate adaptation.

5.5.2 Performance Results

Static networks: We first evaluate the performance using the static traces. Since the channel is usually stable within a transmission time of a frame, the SNR of preambles is used for trace-driven evaluation. This makes it easy to use the same traces to evaluate different frame sizes. We scale the SNR across subcarriers uniformly up and down to get a complete SNR range. In each run, we transmit 400 frames. We multiply the symbols in a frame by the channel coefficients as specified in the CSI trace, and add appropriate amount of Additive White Gaussian Noise (AWGN). The CSI traces are recorded assuming that the noise variance is 1. The frame is then decoded. Based on the scheme being evaluated, the appropriate rate adaptation scheme is called to calculate the appropriate data rate for the next transmission. We use a max retransmission count of 7, after which the frame is dropped.

Figure 5.9 compares our scheme with the existing schemes using 4000-byte frames. This is a common setting for IEEE 802.11n, since frame aggregation is turned on by default in commodity software drivers [8]. On average, our scheme out-performs WiFi by 26%, SOFT by 23%, PPR by 19.5%, SmartNoComb by 3.5%, and SmartUnaware by 19.5%. These numbers indicate: (i) Smart yields the best performance by using both combining and combining-aware rate adaptation, (ii) SmartNoComb is the next best performer due to its partial retransmission-aware rate adaptation, and (iii) WiFi, SOFT, PPR, and SmartUnaware all perform similarly since the rate adaptation tends to pick a rate that allows the transmission to be delivered successfully in one try

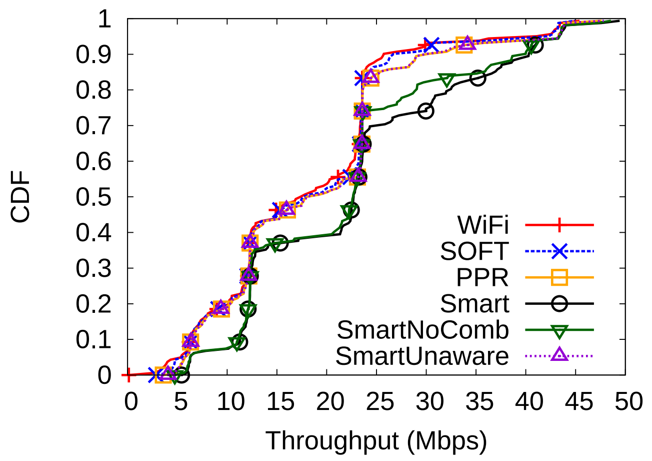


Figure 5.9: Intel static traces with 4000-byte frames: the average throughput of WiFi, SOFT, PPR, Smart, SmartNoComb and 17.7 Mbps, 18.1 Mbps, 18.67 Mbps, 22.31 Mbps, 21.59 Mbps and 18.67 Mbps respectively.

and leaves little opportunity for PPR, SOFT, and SmartUnaware to improve further. Therefore combining-aware rate adaptation is essential to the performance. Furthermore, if we consider the traces where 75% of the frames result in retransmission, Smart shows 15% performance improvement over SmartNoComb. This shows the combining provides significant benefit during retransmissions.

Note that the maximum frame size allowed in 802.11n is even larger: up to 64 KB [66], and the gain of our scheme will increase further with the frame size.

Figure 5.10 compares different schemes using 1000-byte frames. Our schemes continue to perform the best. On average, it out-performs WiFi by 12%, SOFT by 10%, PPR and Unaware combining rate adaptation by 8%, and

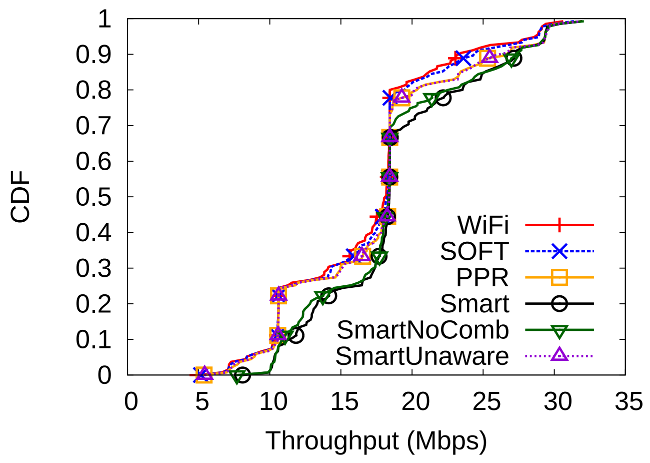


Figure 5.10: Intel static traces with 1000-byte frames: the average throughput of WiFi, SOFT, PPR, SmartNoComb ,SmartUnaware and Smart are 16.72 Mbps, 16.93 Mbps, 17.36 Mbps, 18.38 Mbps, 17.35, and 18.65 Mbps, respectively.

SmartNoComb by 1.5%. The reduced benefit is due to relatively larger MAC overhead for a smaller frame size. Note that the selected rate curves contain a few large jumps. This is because our static traces contain a few subcarriers with very low SNR. Due to such strong frequency selectivity, 1/2 FEC with higher modulation out-performs 3/4 FEC using lower modulation and the 3/4 FEC date rates never get selected.

Mobile networks: Next we compare the performance using the mobile traces. Figure 5.11 shows the throughput under 4000-byte frames. As it shows, on average, our scheme out-performs WiFi by 31.5%, SOFT by 22%, PPR by 13.5%, SmartNoComb by 4.5%, and SmartUnaware by 13%. The performance is consistent with the static traces: Smart continues to perform the best, followed by SmartNoComb. The other four schemes perform similarly due to ineffective rate adaptation. Again, when we consider the traces where 75% of the frames are retransmitted, Smart out-performs SmartNoComb by 12%, which shows the benefit of combining especially during retransmission. Compared with the static traces, the throughput curves of our mobile traces are more smooth since QPSK with 3/4 FEC works sometimes and the rate smoothly changes from QPSK 1/2 FEC to QPSK 3/4 FEC to QAM-16 1/2 FEC.

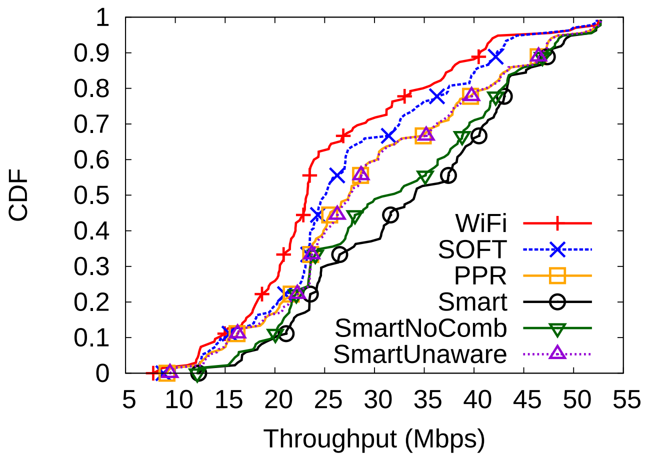


Figure 5.11: Intel mobile traces with 4000-byte frames: the average throughput of WiFi, SOFT, PPR, Smart, SmartNoComb and SmartUnaware are 25.55 Mbps, 27.59 Mbps, 29.62 Mbps, 33.62 Mbps, 32.16 Mbps and 29.76 Mbps respectively.

Figure 5.12 further compares the performance using 1000-byte frames in the mobile traces. On average, Smart out-performs WiFi by 19%, SOFT by 11%, PPR by 6%, SmartNoComb by 2%, and SmartUnaware by 5%. The relative rankings of different schemes remain the same: Smart and SmartNoComb continue to perform well owing to their more effective rate adaptation.

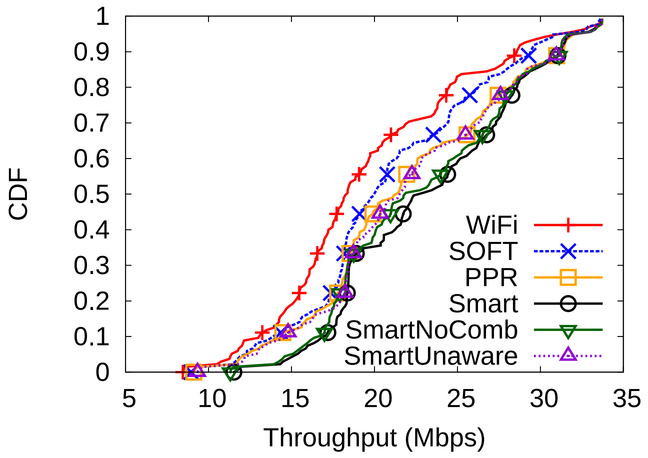
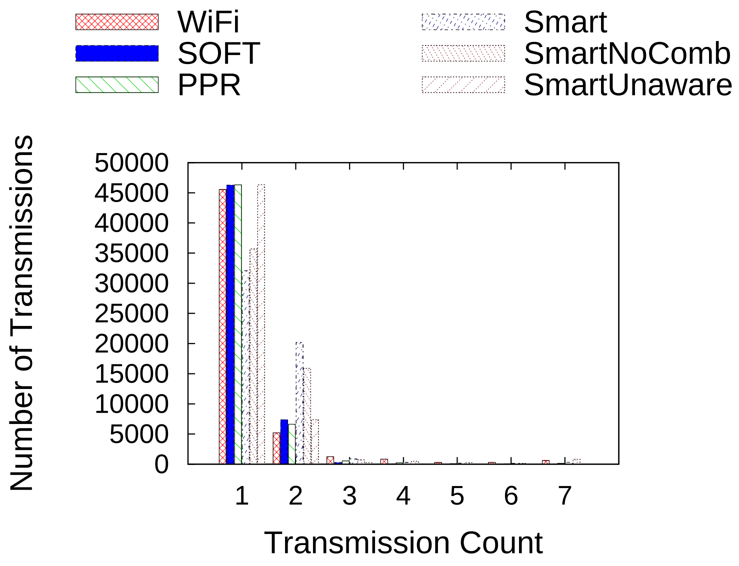
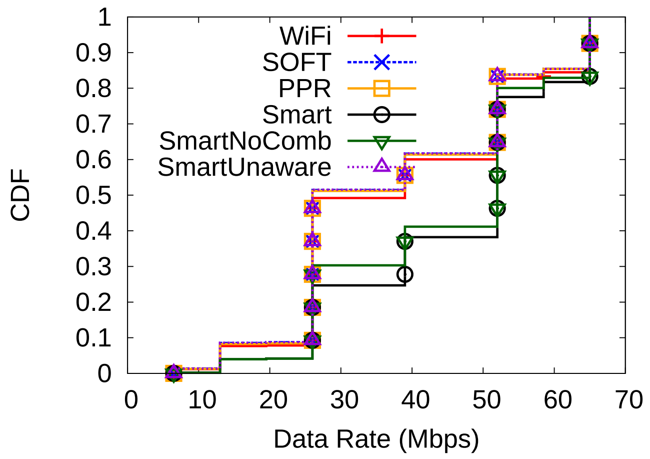


Figure 5.12: Intel mobile traces with 1000-byte frames: the average throughput of WiFi, SOFT, PPR, Smart, SmartNoComb and SmartUnaware are 19.58 Mbps, 20.97 Mbps, 21.94 Mbps, 23.25 Mbps, 22.86 Mbps and 22.13 Mbps respectively.

To further understand the dynamics of different schemes, we examine the number of transmissions and data rates used by each scheme for 4000-byte frames in the mobile traces. Figure 5.13(a) compares the number of transmissions under different schemes. As we would expect, all schemes except SmartNoComb and Smart Retransmission complete most of their transmissions in one try. Specifically, WiFi, SOFT, PPR and SmartUnaware complete 84-86% of the transmissions in 1 try. In comparison, since the rate adaptation in Smart and SmartNoComb are both aware of partial retransmissions and can select a more aggressive rate, they complete 60% and 66% of their transmissions in first try, respectively, and 37% and 29% in two tries. The 6% gap between Smart and SmartNoComb shows that combining allows us to select an even more aggressive rate, thereby achieving higher throughput.



(a) Number of transmissions



(b) CDF of data rates

Figure 5.13: Comparison of numbers of transmissions and data rates in mobile traces using 4000-byte frames.

Figure 5.13(b) further compares the data rates used in different schemes. As we would expect, Smart and SmartNoComb tend to select higher rates, whereas the rates selected by the other four schemes are considerably lower. For example, the median rate selected by Smart is over 50 Mbps, compared to 30 Mbps in WiFi.

Under interference: Next we evaluate the performance under interference. We select a trace of one link as the foreground traffic, and a trace of another link as the interference and let these two transmissions collide randomly with a varying probability. As shown in Figure 5.14, Smart out-performs the other schemes under interference. The throughput gains over WiFi are up to 30%. As before, SmartNoComb continues to perform well due to its effective rate adaptation. The benefit of PPR and SmartUnaware over WiFi increases in the presence of interference because interference allows them to perform partial retransmissions more frequently. In comparison, the performance benefit of SOFT is still limited because it retransmits complete frames even though significant portions of the frames have already been received correctly.

Energy comparison: As mentioned earlier, Smart Retransmission can not only improve throughput, but also reduce energy consumption since it reduces transmission time. We assume the access point (AP) runs the rate adaptation scheme, and quantify the energy consumption of a client that is either transmitting to or receiving from the AP. Figure 5.15(a) and (b) compares the average energy consumption of all schemes using 4000-byte frames in the static and mobile traces, respectively. As we can see, in the static traces, Smart reduces WiFi transmitter energy by 25%, SOFT by 18%, PPR by 15%, SmartNoComb by 1%, and SmartUnaware by 15%. The receiver also shows

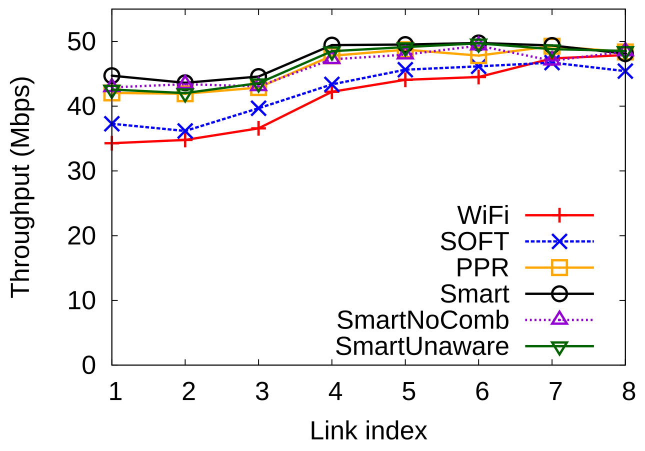
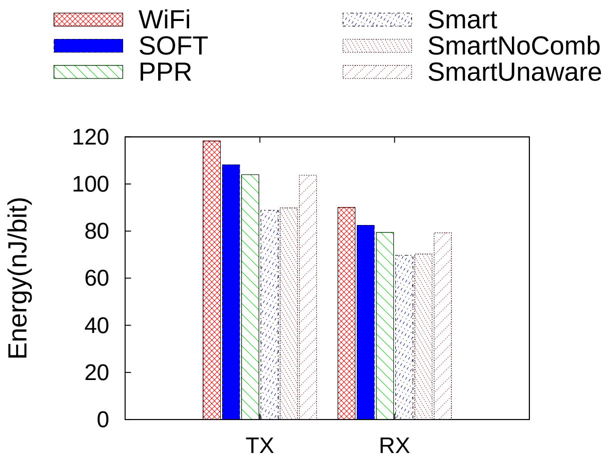


Figure 5.14: Throughput under interference with interference link power scaled down by 0.1 and collision probability is 10%.

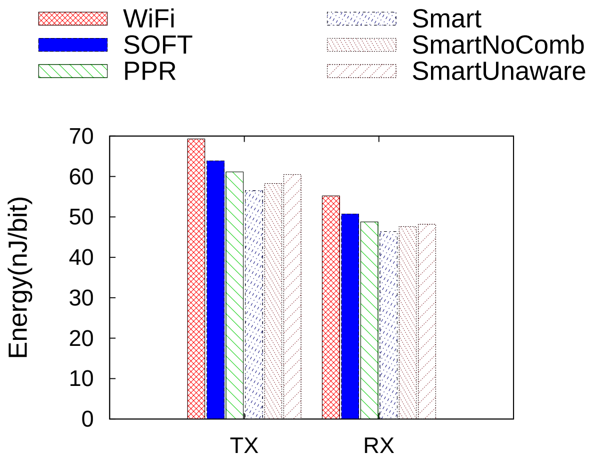
similar energy savings. For mobile trace, the energy savings are 18% over WiFi, 11% over SOFT, 8.5% over PPR, 7% over SmartUnaware and 3% over SmartNoComb. Similarly, the Smart receiver consumes less energy due to reduced reception time.

5.6 Testbed Experiments

Next we evaluate the performance of our scheme using USRP. We have two USRP nodes serve as a transmitter and receiver, and the third USRP node is used to inject narrowband interference. Both flows use 1 MHz channel. The interference allows us to introduce frequency diversity (common in a 20 MHz WiFi channel) to a 1MHz USRP channel. The center frequency of the interfering USRP is selected so that it partially overlaps with the foreground flow for a given number of subcarriers. As in the simulation, we account for



(a) Static traces



(b) Mobile traces

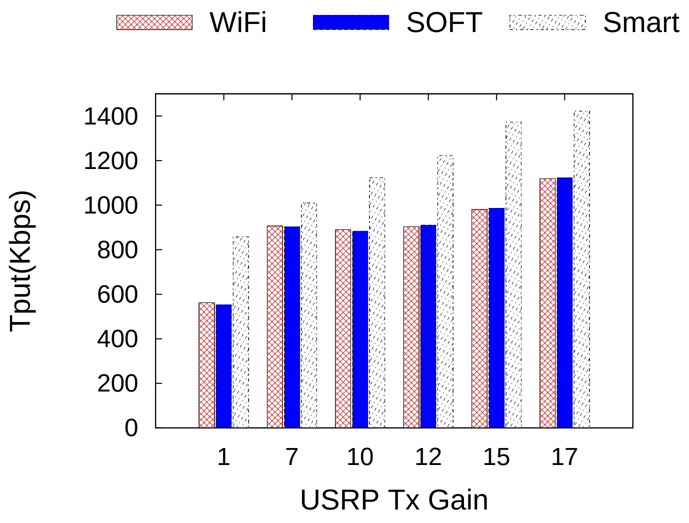
Figure 5.15: Energy consumption using Intel traces with 4000-byte frames.

the feedback overhead in the testbed experiments. We ignore the DIFS/SIFS overhead since they are negligible compared to the data transmission time under 1 MHz channel.

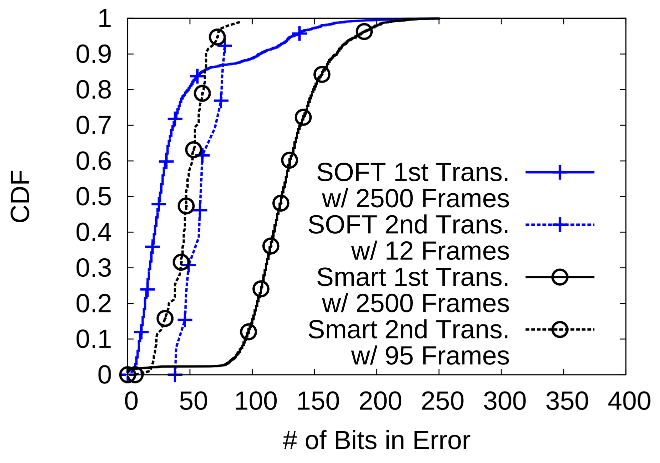
Varying Transmit Power: We first evaluate the performance of our scheme by varying the transmit power of the foreground flow through changing its TX gain. The transmit power and bandwidth of the background interfering node are both fixed. Figure 5.16(a) shows the average throughput of 1000-byte frames using Smart, SOFT, and WiFi as we vary TX gains. For each TX gain value, we average across five runs. Smart outperforms both WiFi and SOFT by 11–52% due to its combining-aware rate adaptation and combining gain. This effect is most prominent for Tx-Gain 15 when both WiFi and SOFT transmit at MCS-3 while Smart transmits at MCS-5 during the first transmission and uses the MCS-7 for retransmission, thereby achieving a much higher throughput. SOFT performs similar to WiFi since the selected rate lets most transmissions succeed in one try and there is little opportunity for SOFT to leverage the combining gain.

Figure 5.16(b) shows the CDF of the number of erroneous bits in a frame after the first and second transmissions for SOFT and Smart under Tx-Gain 15. As mentioned above, Smart selects MCS-5 for the first transmission, which results in more bits in error when compared to SOFT, which selects MCS-3. However, the number of erroneous bits after the second transmission drop quickly in Smart due to its effective combining and rate adaptation. The numbers of erroneous bits in Smart and SOFT do not need to be zero in order for a frame to be delivered successfully due to the use of FEC.

Varying Interference Bandwidth: Next we fix the transmit power and the bandwidth of both transmitters, and vary the amount of the overlapping bandwidth between the foreground and background flows by changing the



(a) Throughput



(b) CDF of number of erroneous bits

Figure 5.16: Throughput under varying transmission power and CDF of number of erroneous bits in 1000-byte frames.

center frequency of the background interference. The center frequency of the foreground flow is 2490MHz, and the center frequency of the background interference varies from 2490.7MHz to 2490.6MHz to 2490.2MHz. These center frequencies correspond to an overlapping region of 30%, 40% and 80%.

Figure 5.17 compares the average throughput of Smart, SOFT, and WiFi for these settings using 1000-byte frames. The throughput is averaged over five runs. Smart again shows significant throughput improvement: it outperforms WiFi and SOFT by 40%, 14% and 36% under these overlap settings, respectively. The improvement is the largest when the number of bad subcarriers is small because Smart needs to retransmit only a few subcarriers. However, even when 80% of the subcarriers have interference, Smart is able to out-perform WiFi by 36% due to its effective rate selection.

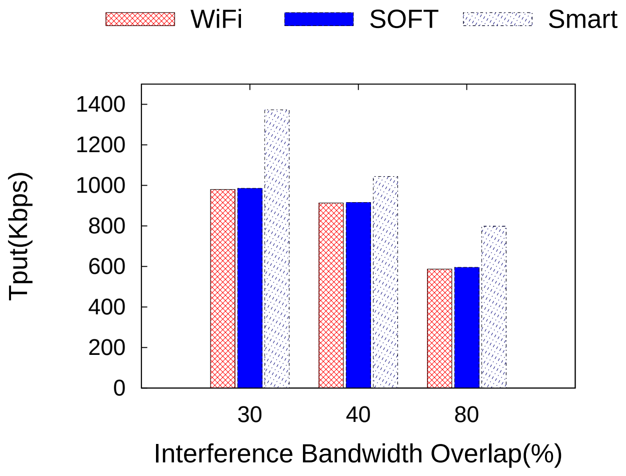


Figure 5.17: Throughput under varying interference bandwidth.

Chapter 6

Conclusion

The proliferation of wireless networks is resulting in an explosive growth of wireless devices and traffic demands. Consequently, it is essential that the protocols used by these devices are efficient and provide good performance.

The goal of this dissertation is to address these problems using rate adaptation. We propose three schemes, which, i) minimize energy consumption, ii) estimate delivery ratio in wireless channels with bursty errors and, iii) maximize throughput using partial retransmission.

For i), we collect and analyze power measurements from different wireless cards and then derive energy models for transmission and reception. Based on the models, we develop a model-driven energy aware rate adaptation scheme. Our energy models show an accuracy error that is consistently below 5%. Furthermore, the simulation and experiments show that our rate adaptation scheme reduces energy by 14-37% over the existing approaches.

In ii), we develop two complementary delivery rate estimation techniques that explicitly take into account the bursty errors and improve estimation accuracy. Our schemes exhibit an average error of 3% and exceed 10% in only 4%–7% of the cases. In comparison, the existing state of the art technique (effective SNR) has an average error of $8 - 17\%$ and has error exceed 10% for 27%–50% of the cases. The improved accuracy of our estimator allows us to more effectively optimize and manage wireless networks. We further develop a new interleaver to reduce the bursty error, and a rate adaptation scheme that incorporates both enhanced delivery rate estimation and interleaver. Our evaluation shows these approaches are effective in improving the predictability, throughput, and energy of wireless networks.

We achieve iii), by developing a combining-aware rate adaptation, which effectively harnesses partial retransmission by identifying the utility in a failed transmission and jointly optimizing with the transmission data rate to minimize the total transmission time for the frame. Our approach shows up to 32% throughput improvement over existing rate adaptation schemes and increases the rate by 1 to 2 steps over the rate selected by the conventional rate adaptation schemes.

Bibliography

[1] LAN/MAN Standards Commmittee of the IEEE Computer Society. Part 11: Wireless LAN Medium Access Control and Physical Layer (PHY) Specifications. *IEEE Standard 802.11*, 2009. `http://standards.ieee.org/getieee802/download/802.11n-2009.pdf`.

[2] Power consumption and energy efficiency comparisons of wlan products. *White Paper, Atheros Comm*, Apr. 2004.

[3] V. Baiamonte and C.-F. Chiasserini. Saving energy during channel contention in 802.11 wlans. *Mob. Netw. Appl.*, 11(2):287–296, April 2006.

[4] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. Energy consumption in mobile phones: A measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '09, pages 280–293, New York, NY, USA, 2009. ACM.

[5] From BCJR to turobo decoding: MAP algorithms made easier. `http://paginas.fe.up.pt/~sam/textos/From%20BCJR%20to%20turbo.pdf`.

[6] BCJR decoding: The BCJR algorithm. `http://www.ii.uib.no/~eirik/INF244/Lectures/Lecture11.pdf`.

[7] G. Benelli. An arq scheme with memory and soft error detectors. *Communications, IEEE Transactions on*, 33(3):285–288, Mar 1985.

[8] Gautam Bhanage, Rajesh Mahindra, Ivan Seskar, and Dipankar Raychaudhuri. Implication of mac frame aggregation on empirical wireless experimentation. In *Proc. of IEEE GLOBECOM*, 2009.

[9] Apurv Bhartia, Yi-Chao Chen, Swati Rallapalli, and Lili Qiu. Harnessing frequency diversity in WiFi networks. In *In Proc. of ACM MobiCom*, Sept. 2011.

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

[11] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Inc., New York, NY, USA, 1995.

[12] R. Bruno, M. Conti, and Enrico Gregori. Optimization of efficiency and energy consumption in p-persistent csma-based wireless lans. *Mobile Computing, IEEE Transactions on*, 1(1):10–31, Jan 2002.

[13] J. Camp and E. Knightly. Modulation rate adaptation in urban and vehicular environments: cross-layer implementation and experimental evaluation. In *Proc. of ACM MobiCom*, 2008.

[14] Aaron Carroll and Gernot Heiser. An analysis of power consumption in a smartphone. In *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*, USENIXATC'10, pages 21–21, Berkeley, CA, USA, 2010. USENIX Association.

[15] Marcelo M. Carvalho, Cintia B. Margi, Katia Obraczka, and J. J. Garcia-Luna-Aceves. Modeling energy consumption in single-hop IEEE 802.11 ad hoc networks. In *Proc. of ICCCN*, Oct. 2004.

[16] D. Chase. Code combining–a maximum-likelihood decoding approach for combining an arbitrary number of noisy packets. *Communications, IEEE Transactions on*, 33(5):385–393, May 1985.

[17] Jyh-Cheng Chen and Kai wen Cheng. Edca/ca: Enhancement of ieee 802.11e edca by contention adaption for energy efficiency. *Wireless Communications, IEEE Transactions on*, 7(8):2866–2870, August 2008.

[18] Cisco visual networking index: Global mobile data traffic forecast update, 20142019. `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html`.

[19] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, December 1989.

[20] Armin Dammann and Ronald Raulefs. Comparison of space-time block coding and cyclic delay diversity for a broadband mobile radio air interface. In *Interface, 6th Int. Symp. Wireless Personal Multimedia Communications (WPMC 2003)*, pages 411–415, 2003.

[21] More than 30 billion devices will wirelessly connect to the internet of everything in 2020. `https://www.abiresearch.com/press/more-than-30-billion-devices-will-wirelessly-conne`.

[22] Fahad R. Dogar, Peter Steenkiste, and Konstantina Papagiannaki. Catnap: exploiting high bandwidth wireless interfaces to save energy for mobile devices. In *Proc. of ACM MobiSys*, 2010.

[23] Mian Dong and Lin Zhong. Self-constructive high-rate system energy modeling for battery-powered mobile systems. In *Proc. of ACM MobiSys*, 2011.

[24] R. Draves, J. Padhye, and B. Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *Proc. of ACM MobiCom*, Sept. - Oct. 2004.

[25] M. Ergen and P. Varaiya. Decomposition of energy consumption in ieee 802.11. In *Communications, 2007. ICC '07. IEEE International Conference on*, pages 403–408, June 2007.

[26] V. Erceg etc. Ieee 802.11 wireless lans: Tgn channel models. 2004.

[27] L.M. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In

INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, volume 3, pages 1548–1557 vol.3, 2001.

[28] Jason Franklin, Damon McCoy, Parisa Tabriz, Vicentiu Neagoe, Jamie Van Randwyk, and Douglas Sicker. Passive data link layer 802.11 wireless device driver fingerprinting. In *Proc. of USENIX Security*, 2006.

[29] A. Garcia-Saavedra, P. Serrano, A. Banchs, and M. Hollick. Energy-efficient fair channel access for ieee 802.11 wlans. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, pages 1–9, June 2011.

[30] Andres Garcia-Saavedra, Pablo Serrano, Albert Banchs, and Giuseppe Bianchi. Energy consumption anatomy of 802.11 devices and its implication on modeling and design. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '12, pages 169–180, New York, NY, USA, 2012. ACM.

[31] H. P. Gavin. The levenberg-marquardt method for nonlinear least squares curve-fitting problems. `http://people.duke.edu/~hpgavin/ce281/lm.pdf`.

[32] Shyamnath Gollakota and Dina Katabi. ZigZag Decoding: Combating

hidden terminals in wireless networks. In *Proc. of ACM SIGCOMM*, 2008.

[33] Mahanth Gowda, Souvik Sen, Romit Roy Choudhury, and Sung-Ju Lee. Cooperative packet recovery in enterprise WLANs. In *Proc. of IEEE INFOCOM*, 2013.

[34] A. Gudipati and S. Katti. Strider: Automatic rate adaptation and collision handling. In *Proc. of ACM SIGCOMM*, 2011.

[35] D. Haccoun and G. Begin. High-rate punctured convolutional codes for Viterbi and sequential decoding. *IEEE Transactions on Communications*, 37(11), 1989.

[36] J. Hagenauer. Rate-compatible punctured convolutional codes (rcpc codes) and their applications. *Communications, IEEE Transactions on*, 36(4):389–400, Apr 1988.

[37] Daniel Halperin, Ben Greensteiny, Anmol Shethy, and David Wetherall. Demystifying 802.11n power consumption. In *Proc. of HOTPOWER*, 2010.

[38] Daniel Halperin, Wenjun Hu, Anmol Sheth, and David Wetherall. Predictable 802.11 packet delivery from wireless channel measurements. In *Proc. of ACM SIGCOMM*, 2010.

[39] Daniel Halperin, Wenjun Hu, Anmol Sheth, and David Wetherall. Tool release: gathering 802.11n traces with channel state information. *SIGCOMM Comput. Commun. Rev.*, 41(1), January 2011.

[40] Hao Han, Yunxin Liu, Guobin Shen, Yongguang Zhang, and Qun Li. Dozyap: power-efficient wi-fi tethering. In *Proc. of ACM MobiSys*, 2012.

[41] B.A. Harvey and S.B. Wicker. Packet combining systems based on the viterbi decoder. *Communications, IEEE Transactions on*, 42(234):1544–1557, Feb 1994.

[42] Simon Haykin. *Neural Networks: A Comprehensive Foundation.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 1998.

[43] G. Holland, N. Vaidya, and P. Bahl. A rate-adaptive MAC protocol for multi-hop wireless networks. In *Proc. of ACM MobiCom*, 2001.

[44] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Netw.*, 4(2):251–257, March 1991.

[45] hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAPRADIUS authenticator. `http://hostap.epitest.fi/hostapd/`.

[46] Intel Ultimate N WiFi Link 5300 and Intel WiFi Link 5100 products. http://www.intel.com/products/wireless/adapters/5000/index.htm.

[47] Data transfer over wireless lan power consumption analysis. *White Paper, Intel Corp*, Feb. 2009.

[48] Internet of everything. `http://www.cisco.com/web/about/ac79/innov/IoE.htm`.

[49] IT++ C++ Library. `http://itpp.sourceforge.net/4.3.1/`.

[50] Kyle Jamieson and Hari Balakrishnan. PPR: partial packet recovery for wireless networks. In *Proc. of SIGCOMM*, 2007.

[51] Ki-Young Jang, Shuai Hao, Anmol Sheth, and Ramesh Govindan. Snooze: Energy Management in 802.11n WLANs. In *Proc. of ACM CoNEXT*, 2011.

[52] Eun-Sun Jung and N.F. Vaidya. An energy efficient mac protocol for wireless lans. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1756–1764 vol.3, 2002.

[53] P. S. Kalekar. Time series forecasting using holt-winters exponential smoothing. Dec. 2004.

[54] Shyamnath Gollakota Kate Lin and Dina Katabi. Random access heterogeneous MIMO networks. In *Proc. of ACM SIGCOMM*, 2011.

[55] Muhammad Owais Khan, Vacha Dave, Yi-Chao Chen, Oliver Jensen, Lili Qiu, Apurv Bhartia, and Swati Rallapalli. Model-driven energy-aware rate adaptation. In *Proc. of ACM MobiHoc*, July 2013.

[56] Hun Seok Kim and Babak Daneshrad. Energy-constrained link adaptation for MIMO OFDM wireless communication systems. *IEEE Transactions on Wireless Communications*, 9, 2010.

[57] H. Krishna and S.D. Morgera. A new error control scheme for hybrid arq systems. *Communications, IEEE Transactions on*, 35(10):981–990, Oct 1987.

[58] C.F. Leanderson and G. Caire. The performance of incremental redundancy schemes based on convolutional codes in the block-fading gaussian collision channel. *Wireless Communications, IEEE Transactions on*, 3(3):843–854, May 2004.

[59] Jungwon Lee, Hui-Ling Lou, D. Toumpakaris, E.W. Jang, and J.M. Cioffi. Transceiver design for mimo wireless systems incorporating hybrid arq. *Communications Magazine, IEEE*, 47(1):32–40, January 2009.

[60] Chi-Yu Li, Chunyi Peng, Songwu Lu, and Xinbing Wang. Energy-based rate adaptation for 802.11n. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, Mobicom '12, pages 341–352, New York, NY, USA, 2012. ACM.

[61] Shu Lin and P.S. Yu. A hybrid ARQ scheme with parity retransmission for error control of satellite channels. *Communications, IEEE Transactions on*, 30(7):1701–1719, July 1982.

[62] Emmanuel Lochin, Anne Fladenmuller, Jean yves Moulin, Serge Fdida, and A Manet. Energy consumption models for ad-hoc mobile terminals. In *In Med-Hoc Net*, 2003.

[63] M. Luby. LT codes. In *Proc. of FOCS*, 2003.

[64] D.M. Mandelbaum. An adaptive-feedback coding scheme using incremental redundancy (corresp.). *Information Theory, IEEE Transactions on*, 20(3):388–389, May 1974.

[65] Justin Manweiler and Romit Roy Choudhury. Avoiding the rush hours: WiFi energy management via traffic isolation. In *Proc. of ACM MobiSys*, 2011.

[66] Meraki white paper: 802.11n technology. *Meraki White Paper*, Feb. 2011.

[67] J.J. Metzner. Improvements in block-retransmission schemes. *Communications, IEEE Transactions on*, 27(2):524–532, Feb 1979.

[68] J.J. Metzner and D. Chang. Efficient selective repeat arq strategies for very noisy and fluctuating channels. *Communications, IEEE Transactions on*, 33(5):409–416, May 1985.

[69] A. K. Miu, H. Balakrishnan, and C. E. Koksal. Improving loss resilience with multi-radio diversity in wireless networks. In *Proc. of ACM MobiCom*, 2005.

[70] S.D. Morgera and V. Oduol. Soft-decision decoding applied to the generalized type-ii hybrid arq scheme. In *Communications, 1988. ICC '88. Digital Technology - Spanning the Universe. Conference Record., IEEE International Conference on*, pages 637–641 vol.2, Jun 1988.

[71] ONOE rate control. `http://madwifi.org/browser/trunk/ath_rate/onoe`.

[72] Jeffrey Pang, Ben Greenstein, Ramakrishna Gummadi, Srinivasan Seshan, and David Wetherall. 802.11 user fingerprinting. In *Proc. of MobiCom*, 2006.

[73] Abhinav Pathak, Abhilash Jindal, Y. Charlie Hu, and Samuel P. Midkiff. What is keeping my phone awake?: characterizing and detecting no-sleep energy bugs in smartphone apps. In *Proc. of ACM MobiSys*, 2012.

[74] PCI EXPRESS X1 to PCI Express Mini interface adapter. http://www.adexelec.com/pciexp.htm.

[75] Ioannis Pefkianakis, Yun Hu, Starsky H.Y. Wong, Hao Yang, and Songwu Lu. MIMO rate adaptation in 802.11n wireless networks. In *Proc. of ACM MobiCom*, 2010.

[76] Trevor Pering, Yuvraj Agarwal, Rajesh Gupta, and Roy Want. Coolspots: Reducing the power consumption of wireless mobile devices

with multiple radio interfaces. In *PROC. ACM/USENIX MOBISYS*, pages 220–232, 2006.

[77] J. Perry, P. A. Iannucci, K. E. Fleming, H. Balakrishnan, and D. Shah. A rateless wireless communication system using spinal codes. In *Proc. of ACM SIGCOMM*, Aug. 2012.

[78] Jean pierre Ebert, Brian Burns, , Adam Wolisz, and Adam Wolisz. A trace-based approach for determining the energy consumption of a wlan network interface. In *In Proc. of European Wireless*, 2002.

[79] Monsoon solutions power monitor. http://www.msoon.com/LabEquipment/PowerMonitor.

[80] D. Qiao, S. Choi, and K. Shin. Goodput analysis and link adaptation for IEEE 802.11a wireless LANs. *IEEE TMC*, Oct. 2002.

[81] Daji Qiao, Sunghyun Choi, Amit Jain, and Kang G. Shin. Miser: An optimal low-energy transmission strategy for. In *In Proc. of the ACM/IEEE Intl. Conference on Mobile Computing and Networking*, pages 161–175, 2003.

[82] Hariharan Shankar Rahul, Swaran Kumar, and Dina Katabi. Jmb: Scaling wireless capacity with user demands. In *Proc. of ACM SIGCOMM*, 2012.

[83] Enrico Rantala, Arto Karppanen, Seppo Granlund, and Pasi Sarolahti. Modeling energy efficiency in wireless internet communication. In

Proceedings of the 1st ACM Workshop on Networking, Systems, and Applications for Mobile Handhelds, MobiHeld '09, pages 67–68, New York, NY, USA, 2009. ACM.

[84] Andrew Rice and Simon Hay. Measuring mobile phone energy consumption for 802.11 wireless networking. *Pervasive and Mobile Computing*, 6(6):593–606, December 2010.

[85] Eric Rozner, Vishnu Navda, Ramachandran Ramjee, and Shravan Rayanchu. Napman: Network-assisted power management for wifi devices. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10, pages 91–106, New York, NY, USA, 2010. ACM.

[86] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly. Opportunistic media access for multirate ad hoc networks. In *Proc. of ACM MOBICOM*, Sept. 2002.

[87] Souvik Sen, Naveen Santhapuri, Romit Roy Choudhury, and Srihari Nelakuditi. Accurate: Constellation based rate estimation in wireless networks. In *Proc. of USENIX NSDI*, 2010.

[88] Wei-Liang Shen, Yu-Chih Tung, Kuang-Che Lee, Kate Ching-Ju Lin, Shyamnath Gollakota, Dina Katabi, and Ming-Syan Chen. Rate adaptation for 802.11 multiuser MIMO networks. In *Proc. of ACM MobiCom*, 2012.

[89] A. Shokrollahi. Raptor codes. *IEEE Trans. Info. Theory*, 2006.

[90] P. Sindhu. Retransmission error control with memory. *Communications, IEEE Transactions on*, 25(5):473–479, May 1977.

[91] E. Uhlemann, L.K. Rasmussen, A. Grant, and P. Wiberg. Optimal incremental-redundancy strategy for type-ii hybrid ARQ. In *Information Theory, 2003. Proceedings. IEEE International Symposium on*, 2003.

[92] USRP. `http://www.ettus.com/`.

[93] A. J. Viterbi. Principles of digital communication and coding. 1979.

[94] Mythili Vutukuru, Hari Balakrishnan, and Kyle Jamieson. Cross-layer wireless bit rate adaptation. In *Proc. of SIGCOMM*, 2009.

[95] Xiaodong Wang, Jun Yin, and Dharma P. Agrawal. Analysis and optimization of the energy efficiency in the 802.11 dcf. *Mob. Netw. Appl.*, 11(2):279–286, April 2006.

[96] S. H.Y. Wong, H. Yang, S. Lu, and V. Bharghavan. Robust rate adaptation in 802.11 wireless networks. In *Proc. of ACM MobiCom*, Sept. 2006.

[97] Starsky H. Y. Wong, Hao Yang, Songwu Lu, and Vaduvur Bharghavan. Robust rate adaptation for 802.11 wireless networks. In *Proc. of ACM MobiCom*, 2006.

[98] Grace Woo, Pouya Kheradpour, Dawei Shen, and Dina Katabi. Beyond the bits: Cooperative packet recovery using physical layer information. In *Proc. of ACM MobiCom*, 2007.

[99] J. M. Wozencraft and M. Horstein. Digitalized communication over two-way channels. In *Fourth London Symp. Inform. Theory, London, UK*, 1960.

[100] J. M. Wozencraft and M. Horstein. Coding for two-way channels. In *Tech. Rep. 383, Res. Lab. Electron., MIT, Cambridge, MA*, 1961.

[101] J. Zhang, K. Tan, J. Zhao, H. Wu, and Y. Zhang. A practical snr-guided rate adaptation. In *Proc. of IEEE INFOCOM*, 2008.

[102] Jiansong Zhang, K. Tan, Jun Zhao, Haitao Wu, and Yongguang Zhang. A practical snr-guided rate adaptation. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, April 2008.

[103] Xinyu Zhang and Kang G. Shin. E-mili: energy-minimizing idle listening in wireless networks. In *Proc. of ACM MobiCom*, 2011.