Improving Packet Delivery Efficiency Using Multi-Radio Diversity in Wireless LANs

by

Allen Ka Lun Miu

S.M., Electrical Engineering and Computer Science,
Massachusetts Institute of Technology (2002)
B.Sc., Electrical Engineering and Computer Science,
University of California at Berkeley (1999)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2006

Author ....
Department of Electrical Engineering and Computer Science
May 26, 2006

Certified by ....
Hari Balakrishnan
Associate Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by ....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Improving Packet Delivery Efficiency
Using Multi-Radio Diversity in Wireless LANs

by
Allen Ka Lun Miu

Submitted to the Department of Electrical Engineering and Computer Science
on May 26, 2006, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

Abstract

Data transmissions in Wireless Local Area Networks (WLANs) often suffer from bit errors that arise from the notoriously complex and time-varying signal propagation characteristics of the wireless medium. A number of physical factors such as attenuation and multi-path are prevalent indoors and can lead to high bit-error rates at the link layer. These in turn lead to packet losses, low throughput, and higher and more variable packet latencies observed at higher layers, degrading the performance of many delay-sensitive and traffic-intensive wireless applications such as games, file-sharing, voice-over-IP, and streaming video.

We use the notion of *path diversity* to develop an approach that improves data delivery efficiency and throughput in presence of transmission errors. Path diversity relies on multiple access points (APs) covering a given area or multiple radios on the user's device (or both). The hypothesis underlying this system is as follows: because frame errors are often path-dependent (e.g., due to multi-path fading), location-dependent (e.g., due to noise), and statistically independent between different transmitting radios, transmissions are likely to succeed from at least one of the available transmitters (transmit diversity). Likewise, multiple radios that all receive versions of the same transmission may together be able to correctly recover a frame, even when any given individual radio is not (receive diversity). Using these principles, we design and implement the *Multi-Radio Diversity (MRD)* system, which leverages the properties of path diversity at the transmitter and receiver to reduce frame loss rates in the link-layer, leading to increased throughput and packet delivery efficiency.

We introduce several techniques that make path selection, retransmission, and rate adaptation work efficiently in a MRD system based on the 802.11 MAC. We used commodity PCs and wireless interfaces to build a MRD system and conducted a wide range of indoor experiments. Our experiments measured throughput gains up to three times over conventional schemes without consuming much extra wireless bandwidth.

Thesis Supervisor: Hari Balakrishnan
Title: Associate Professor of Computer Science and Engineering

To my father and my late mother, Hung-Chow Miu and Rui-Bao Wang,
to my fiancée Jie Zhang,
to my brother and sister-in-law, Kar-Cheong and Eleanor Miu,
and to my nephews Winston and Oliver.
My brother is the first person who taught me
about computers on a
64 KB, 1.023 MHz Apple IIe personal computer
when I was seven years old.

Acknowledgments

This dissertation, and all of my accomplishments during graduate school, could not have been realized without help from a wonderful group of people whom I have had the great fortune of meeting. My deepest gratitude goes to my research adviser, Hari Balakrishnan, for instilling confidence in me and for guiding me back on track whenever I felt lost. I am also very grateful for Hari's time and patience in giving detailed feedback on the many revisions of this dissertation, as well as for all of my prior papers and presentations.

Victor Bahl, John Apostolopoulos, and John Ankcorn have been excellent collaborators and caring mentors throughout my graduate career. Thank you for sharing your invaluable advice and life experience with me.

I thank Dina Katabi and Robert Morris for serving on my thesis committee and for their insightful suggestions that help greatly improve the quality of this dissertation. I thank Godfrey Tan, Kyle Jamieson, John Bicket, Sanjit Biswas, Doug Decouto, Wenjun Hu, Eugene Shih, Daniel Aguayo, John Guttag, Dan Tan, and Mitchell Trott for sharing their ideas and comments on my work.

Godfrey Tan was instrumental in the implementation of the early versions of Divert, which has been folded into the transmit diversity sub-system presented in this dissertation. Can Emre Koksal contributed the analysis of the frame combining algorithm and the capacity analysis of MRD. It has been a pleasure collaborating with and learning from these very smart colleagues.

Besides this dissertation, I also had the privilege of working on two other cool research projects at MIT: Cartel and Cricket. To the Cartel team (Vladimir Bychkovsky, Kevin Chen, Michel Goraczko, Hongyi Hu, Bret Hull, Samuel Madden, Eugene Shih, Yang Zhang): it's been a fun ride with you guys (no pun intended). To the Cricket team (Bodhi Priyantha, Michel Goraczko, Dorothy Curtis, Ken Steele, Adam Smith, David Moore, Arvind Thiagarajan, Roshan Baliga, Nikos Michalakis, Kevin Wang, Kalpak Kothari, and Rafael Nogueras, Seth Teller, Erik Demaine, and Steve Garland): it was hugely enjoyable working with each and every one of you.

My time spent at MIT would not have been nearly as memorable without my awesome officemates in G930: Daniel Abadi (we finally got rid of the cactus), Magdalena Balazinska (Mrs. Margaret has a strategy for everything), Jennifer Carlisle (thanks for all of the chit-chats and showing me your palming talent), Kyle Jamieson (the nicest person I know, bar none), Asfandyar Qureshi (thanks for coming in at odd hours and being my company), and Godfrey Tan (who is full of crazy ideas and opinions. Thanks for being a great friend and for always inspiring a positive outlook on every obstacle in life.).

And then there is Room G920, where I can always go to and find someone to talk with. Bret (has good taste and a great sense of quick-witted humor), Eugene (knows a lot of cool stuff and is a great sounding board for bouncing ideas), Vlad (strives to be perfect): thanks for being lunch buddies, and for all of the great jokes and the many intelligent and un-intelligent conversations we've had. I also thank my Room 512 ex-officemates, Dave Andersen, who was really supportive and just incredible all-around, and Alex Snoeren, who showed me the ropes around MIT when I first arrived. Life in the lab couldn't have been as fun and stimulating without Ali Shoeb, Hariharan Rahul, Jaeyeon Jung, Jayashree Subramanian, Karen Zee, Lewis Girod, Manish Bhardwaj, Mike Walfish, Mythili Vutukuru, Nate Kushman, Sachin Katti, Srikanth Kandula, and Stanislav Rost. I wish to express a great appreciation for Michel Goraczko, who not only helped me with various Linux problems but also gave support and thoughtful words when I needed them the most. And while making sure that administrative things do not get in the way of our daily activities, Sheila Marian has indulged all of us with her delicious and sinful home-made desserts. I'm going to miss you all very much!

The companionship from all of my friends makes me feel like the luckiest person on the planet. There are too many of them to thank but a select few (and yes, the following list represents only *a few*) deserve special recognition. Norman Lee (my best-man, my best-friend since high school, and my great badminton partner with a vicious fore-hand) simply gives without expecting anything in return. I couldn't ask anything more from a friend like you. Without the help from Yvonne Lee and Brandon Lai, my wedding proposal plan (conceived during an experiment in the transmit diversity section) would not have succeeded. I thank Jookun Lim and Vivian Tam for their many home-cooked meals and for inviting me over for the holidays. Alec Woo, Rona Yang, Victor Wen, Wen Huang were there to share many good times, and talk about our ambitions. Winnie Ng, Carrie Ng, Tim Haaf, and Billy Huang: I appreciate how we remain close friends after all these years being apart. Finally, life outside of work would not have been the same without all of the awesome folks from the MIT Badminton Club: Archana Venkataraman, Austin Che, Bel Pituwong (heart-break kid), Camille and Wesley Pan, Chia-ying Chu, Chun-Yuan Hsiao, Chris Lawrence (Boston Open), Cindy Wang, Dr. Todd Brennan, (Little) Feng Xia, Greg and Stephanie Dancer, Ivan and Jenny Siew, Jennifer Huang, Jeremy and Gabriel Siew, Kwan Meng Wei (Decent-Meng), Marjan Rafat, Mary Presley, Norman Margolus, Ronian Siew, Uncle Tam, Wen and Kuan Wang, and many, many, many other nice people whom I have left out but were a big part of enriching my life and keeping me sane and in good shape at MIT.

I dedicate this work to my family and my fiancée Jie Zhang for their unconditional love, support, and encouragement. My late mother taught me how to use my brain to work through tough problems and to never give up. My father taught me the values of hard work, persistence and dependability. I thank my brother for sparking my insatiable curiosity in computers at a very young age and for setting a great example for his little brother. I'm grateful for "Ah E" and Auntie Iu for their care and support, and putting a big smile on my face whenever I pick up the phone and hear their voice or receive the many red envelopes that they've sent me over the mail. Margaret and Cecil Haaf also deserve a special thank you for their kindness and loving care since childhood. I'm thankful for Mary Shu and John Zhang, who treated me and supported me like their own son from the very beginning when I first dated their daughter. Jie Zhang is my soul-mate and my source of drive and inspiration. She sacrificed her own needs and encouraged me to attend MIT to pursue my dreams even though it meant separating us physically apart on the different coasts of the country. Nonetheless, she has given me relentless support, warmth, comfort, and love all along. It looks like we have finally made it to the end of a very long journey and are ready to embark on a new one!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Wireless local area networks (WLANs) have gained great popularity in homes, offices, shopping malls, and airports, providing tetherless, high-speed connectivity to the Internet. According to In-Stat, between 110 and 140 million 802.11 WLAN chipsets have shipped in 2005 and the annual shipment number is expected to reach 430 million by 2009 [10]. A growing number of users connect to the Internet using WLANs for emails, instant messaging, file transfers, and web browsing. Increasingly, consumer electronic devices such as digital photo cameras, and music players are being incorporated with WLAN interfaces to help them share data across the network. Specialized WLAN communication devices such as mass storage, portable voice-over-IP (VoIP) phones, video-conferencing stations, gaming consoles, and television have also emerged. Examples of future WLAN applications may include telemedicine and virtual reality.

Many of these exciting applications demand fast and reliable network performance. Current WLAN devices such as 802.11a [14] and 802.11g [16] are capable of delivering data at high raw bit rates up to 54 Mbits/s. However, the end-to-end performance achieved in today's WLANs is often inconsistent and is measured, as we will show in this dissertation, to be far lower than the highest achievable rate.

Figure 1-1 shows some evidence of this poor and inconsistent performance for an experiment (described in Section 3.6.2) that used a moving transmitter and a stationary receiver to measure the throughput and the frame loss rates of a conventional 802.11a link. Although the distance between the two terminals is relatively short ($\approx$6 m), and the dimension of the area in which the transmitter moves is small (about 1.5 m $\times$ 2 m), the measured throughput fluctuates by almost an order of magnitude between 2 and 18 Mbits/s. The average throughput (8.25 Mbits/s) shown in this experiment is far from the highest achievable rate in 802.11a ($\approx$31 Mbits/s, after various overheads are removed).

A significant cause for poor performance is data corruption during transmission over the wireless medium. Wireless communication channels have notoriously *time-varying* characteristics, where the quality of received signals changes dramatically even over time durations lasting just milliseconds. The complex behavior of wireless signal propagation, particularly indoors, is due to five main causes: *noise* at the receiver, typically caused by both external sources and thermal energy in the electronic components at the receiver; *attenuation*, caused both by distance from the transmitter and by stationary or moving obstacles shadowing the signal's path to the receiver; *interference* from other transmitters; *multipath* signal propagation that causes unwanted reflections and distorts reception; and *user mobility*, which causes rapid channel variations. These properties lead to transmission errors at the link

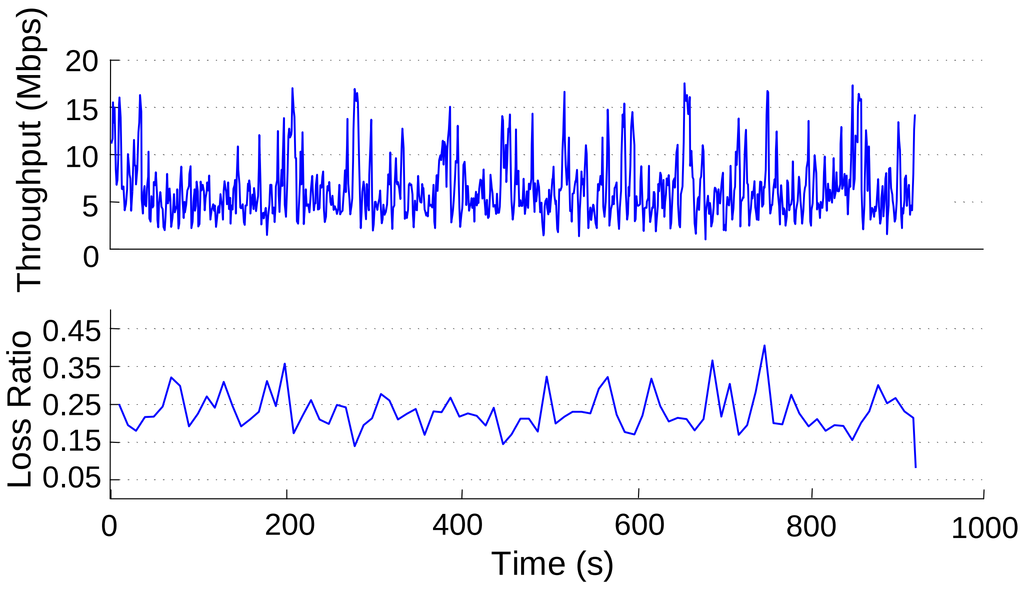


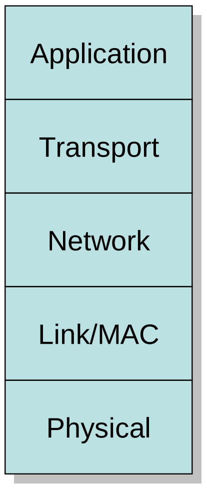
Figure 1-1: An example of poor performance in an 802.11a experiment that involved a moving transmitter and a stationary receiver. The top graph shows how throughput can fluctuate by almost an order of magnitude as the transmitter moves within a small 1.5 m × 2 m area (Figure 3-8). The bottom graph shows the corresponding frame loss rates averaged over one-second windows.

layer, which in turn results in packet losses, low throughput, and higher and more variable packet latencies at higher layers.

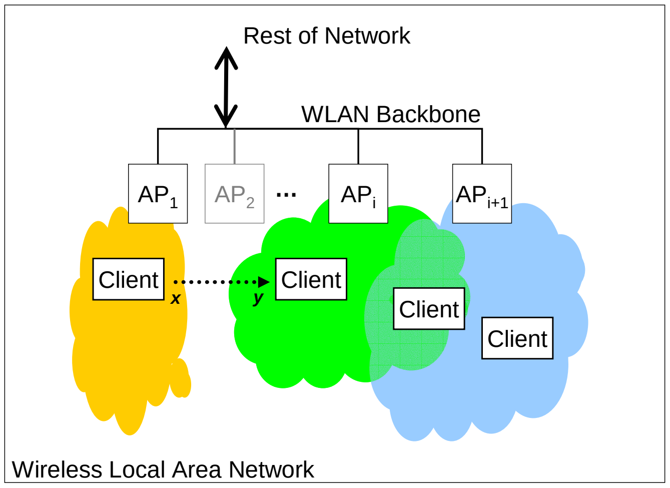
The goal of this dissertation is to take a fresh approach of developing techniques that uses *multiple* radios to help reduce transmission errors and increase the efficiency of delivering packets in a wireless LAN. Using a wide range of experiments performed on an indoor 802.11a testbed, we show that our system can achieve throughput improvements by up to a factor of 3× over the conventional system. The rest of this chapter is organized as follows: Section 1.1 reviews the basic components and structure of a wireless LAN. Section 1.2 motivates our thesis of using multiple radios in wireless LANs to improve performance. Section 1.3 presents an overview of the system we developed that uses multiple radios to improve WLAN performance. We summarize our contributions and lay the roadmap for the rest of the dissertation in Section 1.4.

1.1 Background

Delivering information from one host to another over a computer network is a complex task. To manage this complexity, network designers have subdivided networking systems into *layers*, each of which performs different network functions that are needed to deliver a piece of information to the intended destination in the intended manner specified by a *network protocol*. Layering helps manage complexity because the interface between any two layers specifies a *limited* set of services that the lower layer needs to provide to the higher layer and *hides* from the higher layer the design and implementation details of the lower layer. There are numerous good references that give a detailed treatment on the designs of



(a) Network Layers.



(b) Model of today's WLAN.

Figure 1-2: Left: A five-layer reference model that defines different functions that a network performs to move information from one host to another. of a set of clients and one or more interconnected access points. There is no centralized control over routing information so the client is responsible for initiating association and disassociation to maintain connection and proper routing within the network. As an example, when a client moves from location x to y, it dissociates with AP_j and associates with AP_{j+1}

computer networks [74, 22, 88] and on wireless networks [78].

A convenient reference model for the work described in this dissertation is the five-layer model shown in Figure 1-2(a). Host applications that generate data for network delivery belong to the *application layer*. They use a *transport layer* protocol, such as the *Transport Control Protocol (TCP)* or the *User Datagram Protocol (UDP)*, to transfer data to the application running on the destination host. The transport layer provides a virtual connection that transfers data between two end applications and hides the details of (1) fragmenting a byte stream into smaller, discrete units called *packets*, (2) delivering data to the intended application (*e.g.,* via port numbers), (3) adapting the data transfer rate to the time-varying capacity of the path (*e.g.,* via congestion control), and (4) delivering data reliably to the destination (*e.g.,* via retransmissions).

Protocols at the transport layer use the *network layer*, *e.g.,* the *Internet Protocol (IP)*, to forward packets to the destination without having to worry about the details of forwarding packets through intermediate nodes in a network. The network layer uses the *link layer* (also known as the *Medium Access Control (MAC)* layer) to transmit a packet over a link between two nodes. The link layer abstracts the details of the underlying technology of a link (*e.g.,* a wireless link), and the details of how to use a given link most efficiently to deliver a packet over the link. Finally, the *physical layer* provides a "bit-pipe" service that converts the data bits from the link layer to analog signals that are transmitted over the physical medium. This dissertation examines the properties of the physical layer and uses these properties to improve the packet delivery efficiency of the WLAN link layer.

Figure 1-2(b) shows the architecture of a WLAN, which consists of a set of wireless client terminals and one or more *access points* (APs). APs are interconnected (usually wired) network elements that relay network layer packets between the client terminal and the rest of the network. APs and wireless terminals encapsulate an entire or a fragment of a network layer packet into link layer transmission units called *frames*. A single packet may be transmitted over the wireless medium using several different frames, depending on the level of fragmentation and the number of retransmission attempts used to transmit the packet.

At the link layer, the sender transmits a frame by passing it to the physical layer, which then encodes and modulates the data bits into a radio signal. The receiver reverses the process of demodulating, decoding the received signal into a link layer frame, and decapsulating the frame(s) to assemble the original network layer packet.

When a packet destined to a client arrives, the infrastructure needs a way of knowing which AP it should use to relay the *downlink* packet to the client over the wireless medium. Similarly, in the other *uplink* direction, the client needs to pick among multiple possible APs. Because each AP in a conventional WLAN operates independently without any central coordination, the clients become responsible for discovering, selecting, and associating with an AP in the network. The process of discovering and selecting APs involves *scanning* a finite number of radio channels to identify the AP with the best signal quality within its radio range. The process of *association* entails switching the client's radio channel to the same one used by the targeted AP, and registering with the AP so that it can start relaying packets for the client. As a client begins to move out of range of an AP (*e.g.,* moving from location x to y in Figure 1-2(b)), it scans for alternate APs within its range. When the client finds another AP that gives a good signal quality at the new location, it *roams* or performs a *handoff* to the new AP by issuing a disassociation message to the old AP and associating with the new one. In current 802.11 WLANs [15], the handoff process usually takes several tens to hundreds of milliseconds [63].[1]

Even when a client is associated with an AP that gives good signal quality, frame loss rates can still be significant due to *collisions* and transmission errors. Collisions happen when multiple senders transmit at the same time and their transmissions interfere with each others' receptions. Most WLANs adopt a widely used technique called *carrier sense multiple access (CSMA)* to alleviate collisions in a network where terminals access the channel in an uncoordinated manner. In CSMA, a terminal that needs to access the channel first picks a random backoff delay in which the terminal listens to the channel for a possible ongoing transmission. If no transmission is detected within the backoff delay, the terminal begins its transmission. Otherwise, the terminal defers its transmission until the channel becomes clear to avoid collision. The range of the random backoff delay is bounded by a *backoff window*. The size of the backoff window affects the collision rate and the frame's transmission delay and is adjusted based on the level of utilization in the channel. WLANs such as 802.11 conservatively assume that all frame losses are caused by collisions and indicate a high level of contention. Thus, WLAN terminals increase the backoff window exponentially on every failed transmission attempt to avoid collisions.

However, not all losses are caused by collisions. Frame losses can often arise from transmission errors inflicted by the physical causes described earlier. Because detrimental effects from physical causes are prevalent and degrades communication throughput and latency, an important requirement in the design of high performance WLAN systems is loss resilience.

Previous work over the past several decades has led to a number of error correction and

[1]In comparison, the time required to transmit a 1500 byte frame at a bit-rate of 11 Mbits/s in an 802.11b network is only about 2 ms.

loss recovery strategies. The simpliest variant is *Automatic-Repeat-reQuest* (ARQ) [62], which applies an *error-detection code* (*e.g.,* Cyclic Redundancy Codes [88]) on every frame. The sender retransmits frames with bit-errors detected at the receiver. This technique is effective as long as the transmitter does not exhaust the retransmission limit, but can be wasteful because entire frames need to be transmitted even when the receiver only needs to recover only a few corrupt bits. Also, delay increases with the number of retransmissions. For WLANs that use CSMA (*e.g.,* 802.11), the delay is further exacerbated by a backoff window that increases exponentially with every retransmission attempt.

An alternate coding technique is forward error correction (FEC) [61]. In FEC, the sender encodes data with extra bits above the physical layer to help the receiver decode the original transmission in presence of errors. Depending on the encoding scheme and the number of extra bits being encoded, FEC can recover from a limited number of bit corruptions. To improve loss resilience when the bit-error rate is high, the transmitter can modulate the data using a lower bit-rate at the physical layer [78, 15]. At low bit-rates, the transmitter essentially uses more signal energy to encode each bit of data and helps the physical layer at the receiver to match the incoming signal to the correct data symbol. The increased robustness of the transmission comes at the expense of reduced throughput. For efficiency, both FEC and physical layer modulation techniques need to avoid encoding too many redundant bits or slowing bit-rates excessively for a given channel. There is a large body of work devoted to the development of FEC or bit-rate selection algorithms that adapts to varying channel conditions [61, 51, 44, 55, 76, 80, 24]. However, the large channel variations that are commonly found in the wireless medium pose a significant challenge in developing efficient and practical adaptive algorithms. Current algorithms still incur significant overhead and do not work well in many environment where high variations exist, especially when nodes are moving.

1.2 The Case for Path Diversity

This dissertation develops an approach to improve error resilience against the *physical causes of transmission errors* in WLANs using the notion of *path diversity.* Path diversity relies on multiple access points (APs) covering a given area or multiple radios on the user's device or a combination of both. The hypothesis underlying this system is as follows: because frame losses are often path-dependent (*e.g.,* due to multi-path fading), location-dependent (*e.g.,* due to noise), and statistically independent (*e.g.,* due to thermal noise in the electronics) between different radios, transmissions are likely to succeed from at least one of the available transmitters (transmit diversity). Likewise, multiple radios that all receive versions of the same transmission may together be able to correctly recover a frame, even when any given individual radio is not (receive diversity). Using these principles, we design and implement the *Multi-Radio Diversity (MRD)* system, which leverages the properties of path diversity at the transmitter and receiver to reduce frame loss rates in the link layer, leading to increased packet delivery efficiency and throughput for higher layer protocols.

MRD is different from today's WLAN architecture, in which a client treats each AP and radio as an independent entity and limits communication to a single path at a time. In contrast, with MRD, a client treats APs and radios attached to the same infrastructure or to the same device as collective entities that can help each other to improve frame delivery rates in the network. Thus, MRD can exploit the physical properties of path diversity and opportunistically make use of all available radios and APs within range to increase the

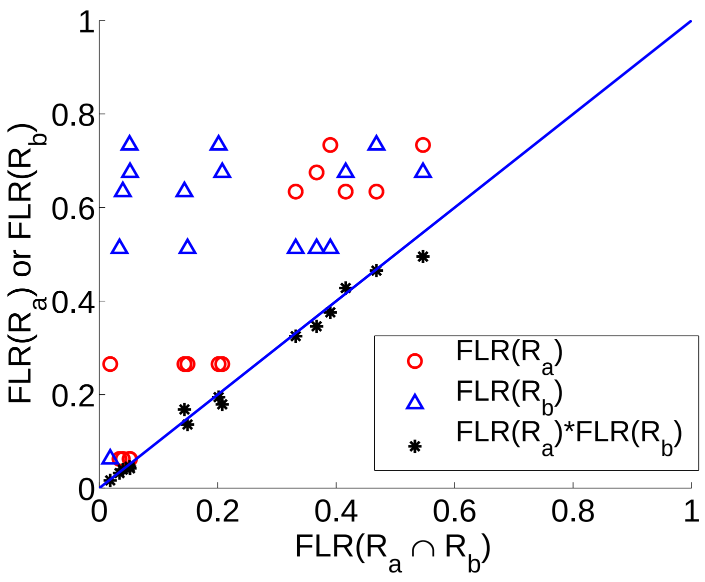


Figure 1-3: Experimental results that show independent loss behavior between all pairwise combinations of six simultaneous receivers.

efficiency of delivering packets in the network.

1.2.1 Loss Independence

The effectiveness of path diversity critically depends on loss independence: multiple radios provide little help if losses are highly correlated among them. Indeed, loss independence may not be valid under certain conditions, such as any interference that arises from an active source (*e.g.,* a microwave oven) that simultaneously affects multiple APs and radios within a given area.

We present measurements that show that a good fraction of transmission errors arise from *path-dependent* physical effects. In one experiment, a sender broadcast frames to six (potential) receivers over an 802.11a network. The receivers were located at different places on the same floor of an office building. The sender sent 500,000 1500-byte User Datagram Protocol (UDP) packets, each tagged with an unique sequence number, at an offered load of 30 Mbits/s over a 48 Mbits/s link. We then took each pairwise combination of the six receivers (a, b) and for each receiver pair, computed the frame loss rate (FLR) at R_a and R_b, and their simultaneous loss rate ($FLR(R_a \cap R_b)$), *i.e.,* the likelihood of both receivers simultaneously observing a loss of the same transmitted frame.

Figure 1-3 plots the individual FLR for R_a (circle) and R_b (triangle) vs. the simultaneous FLR for the R_a and R_b pair.[2] The plot shows that the individual FLR values are greater than the simultaneous FLR, which suggests that losses are not completely correlated between any two receivers. As a result, path diversity may be used to reduce losses compared to a system that uses only a single path.

To test for loss independence, we plot the product of the individual FLRs (star) vs. the simultaneous FLR. The plot shows that the product values lie roughly on the $y = x$ line,

[2]Circles and triangles that lie on the same x-axis value belong to one R_a and R_b receiver-pair. There are $\binom{6}{2} = 15$ receiver pairs in all.

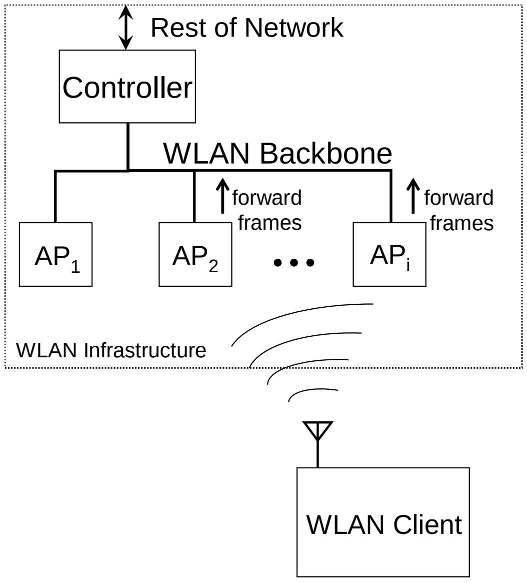


Figure 1-4: With receive diversity, the WLAN infrastructure can improve performance by using multiple APs to receive transmissions from any given client. The infrastructure uses a central controller to filter redundant receptions and to handle retransmissions and in-order packet delivery.

indicating that $FLR(R_a)FLR(R_b) \approx FLR(R_a \cap R_b)$. That is, losses are largely independent at each receiver in this experiment. Another similar wireless measurement study observes loss independence between simultaneous receivers too [89]. We show more evidence of loss independence at the bit-level, and also for transmissions between two alternating transmitters later in Chapter 4.

1.3 Using Path Diversity in WLANs: An Overview

We just presented some empirical evidence that shows that path diversity exists between receivers and Chapter 4 will present evidence that supports path diversity exists between transmitters as well. The properties of path diversity are different between the receiving and the transmitting end of the communication but diversity at either or at both ends may be used to help reduce losses in the wireless link. We outline the challenges and our methods of actually making use of them below.

1.3.1 Receive Diversity

In receive diversity, different APs with overlapping coverage, listening on the same radio frequency, provide alternate communication paths for each frame transmission from a given WLAN client (Figure 1-4), while multiple wireless cards on the WLAN client achieve the same result for transmissions to the client. A central controller collects all received versions of the same transmission, and forwards one of those that has been received correctly.[3]

[3]Thus, APs in MRD no longer relay *packets*. Instead, they relay link layer frames to a central controller, which performs the frame decapsulation task that was originally performed at the AP in the conventional

Although loss resilience may improve as the number of available receivers increases, transmissions can still fail when *none* of the received versions of the transmission are correct. Such failures lead to lost packet transmissions, and packet loss rates of even a few percent can severely degrade the throughput of transport protocols such as TCP. While it is impossible to guarantee reliable transmissions in the wireless medium, our system incorporates three *receive diversity* techniques to reduce packet losses efficiently in the presence of transmission failures:

1. *Frame combining:* MRD attempts to "combine" the different erroneously received versions of the transmitted frame to recover the correct version of the frame. This approach to frame combining is reminiscent of an old, well-studied idea called "retransmissions with memory" [83, 30], where retransmissions of erroneous frames are combined with the original transmission in an attempt to recover the correct version of the data. The computational complexity of this technique is exponential in the number of bit values that are different between the different received versions of the transmitted frame. Our contributions are an generalizing this idea to incorporate the spatial dimension, and making it computationally feasible by using a practical block-based heuristic.

2. *Retransmission using request for acknowledgments:* MRD can often recover a corrupt frame without requiring a retransmission from the client, but frame combining will not always succeed. MRD uses a lightweight retransmission scheme running in between the network layer and the WLAN link layer to further improve error recovery. To prevent adverse interactions caused by the retransmission schemes at two different layers, MRD turns off link layer retransmissions but retains the link layer feedback mechanism for controlling wireless channel contention (congestion). Because some frames can only be recovered after frame combining, MRD uses a feedback protocol between the sender and the central controller where frame combining is performed. This protocol is designed to have low overhead, using a *Request for ACK (RFA)* technique rather than traditional acknowledgments (ACKs) or negative acknowledgments (NACKs), and ensures in-order packet delivery for the higher networking layers.

3. *MRD-aware rate adaptation:* Neither frame combining nor retransmissions can sufficiently cope with communication channels that deteriorates severely. In such environments, the transmitter must use a more robust, but lower bit-rate, modulation scheme to maintain communication. Current bit-rate selection algorithms behave sub-optimally under MRD because they do not use information observed at *all* of the AP and radio receivers that are within range of the sender. We modified an existing bit-rate selection algorithm to adapt bit-rates according to the feedback information from the RFA protocol.

The combination of these techniques forms the MRD-Receive Diversity (MRD-RD) subsystem. A noteworthy aspect of MRD-RD is that it achieves significant improvements in loss rates—especially in highly variable channel conditions that traditional retransmission and bit-rate selection schemes have difficulty coping with—while consuming only a small amount of additional bandwidth. Chapter 3 describes the design and implementation of MRD-RD. We study the performance of MRD-RD using a wide-range of experiments including both

WLAN.

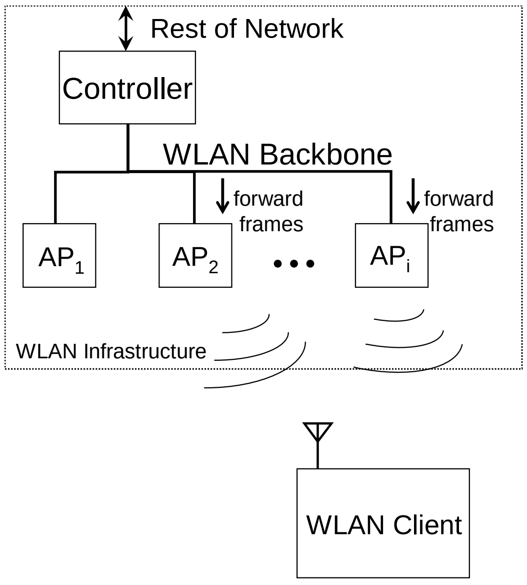


Figure 1-5: With transmit diversity, the WLAN infrastructure can improve performance by using the central controller to track link conditions at each AP and select a good AP for each frame transmission.

stationary and moving wireless receiver nodes on an indoor 802.11a testbed. We find that it can achieve throughput gains of up to 3× over single-path communication schemes that employ a standard 802.11 bit-rate selection algorithm.

1.3.2 Transmit Diversity

As we will show in Chapter 4, frame losses often occur in bursts within a single transmission path, and many of these bursts are of long lengths on the order of tens of frames. Such long burst lengths imply that the conditional probability of losing a frame given that the previous one had been lost is often significantly larger than the average frame loss rate. At the same time, losses are not highly correlated along different transmission paths. As a result, a good choice of transmission points (among a set of APs, as shown in Figure 1-5) or among multiple wireless interfaces installed on a client device) can significantly improve performance.

Our goal is to avoid frame losses by selecting transmission paths adaptively for every transmission. Transmit diversity is especially useful for scenarios when only one radio is used for reception (due to physical or energy limitations on the client's device or the range of the transmission does not reach multiple radios). To implement a practical transmit diversity scheme in WLANs, we had to overcome the following challenges:

1. *High path switching cost:* To adapt to short-term variations in the channel, the process of switching transmission paths to and from a client needs to occur within milliseconds. Current WLANs cannot support this because routing depends on association between an AP and a client. The process of association usually entails AP scanning and a sequence of message exchanges used to authenticate and register routing information for a client. To avoid the high path switching overhead, our system uses a central

controller to manage routing so that switching paths no longer requires negotiation between the APs and the client.

2. *Rapid channel variations:* The state of the channel can change drastically within tens of milliseconds, which makes it difficult to predict the best transmission path at any given moment. Furthermore, the path switching algorithm needs to obtain channel measurements in order to select paths adaptively. However, obtaining precise channel measurements from all of the available transmission points is impractical because it entails sending probes from each of them. Thus, a practical path selection algorithm must deal with both high channel variations and limited channel information.

We developed MRD-Transmit Diversity (MRD-TD), a multi-radio diversity WLAN subsystem that decouples the process of associating a client with an AP from the process of delivering data frames to the client to facilitate low-overhead path switching, and incorporates a practical fine-grained path selection heuristic that effectively reduces burst losses in environments with high channel variations. MRD-TD runs in conjunction with a longer-term primary-AP selection mechanism, usually a card-specific proprietary mechanism, and can also be used with techniques for coping with high frame loss rates such as packet fragmentation [90], varying packet size [69], forward error correction (FEC), bit-rate adaptation [44], and other mechanisms for improving performance in multi-rate WLANs [80, 87].

We conducted experiments on a 802.11b testbed with two APs and a laptop receiver at three different indoor locations, measuring loss rates for two scenarios: one with the laptop receiver moving while the experiment was being conducted, and another with the laptop staying stationary. Our results show that the prototype MRD-TD system reduces the average frame loss rates by as much as 26% compared to a fixed-path scheme that uses the best available path when receiver is in motion. MRD-TD also improves the transmission delay distribution by avoiding long burst losses. Chapter 4 describes the design and implementation of MRD-TD and presents the results of several experiments.

1.3.3 Receive+Transmit Diversity

We combine the previous two sub-systems into a single system to provide diversity gains in both communication directions. For clients that have only a single radio, a MRD WLAN can use MRD-RD to perform frame combining in the uplink direction (from the client to the APs), and MRD-TD to perform path selection among the APs in the downlink direction.

For clients that have multiple wireless interfaces, we can run both MRD-RD and MRD-TD to provide both receive and transmit diversity gains in the same direction of traffic. For example, in the downlink direction, MRD can provide receive diversity gains using multiple receive radios installed on the client and provide diversity gains from multiple APs in the infrastructure. We present the design and implementation of an integrated MRD-Transmit/Receive Diversity (MRD-TRD) system and evaluate its performance gains over the individual sub-systems. Our experiments show that the combined MRD-TRD scheme provides up to 34% and 12% throughput gain over schemes over MRD-TD and MRD-RD, respectively.

1.4 Summary of Contributions and Roadmap

This dissertation presents four main contributions:

1. **Design and implementation of MRD:** We show how to incorporate and integrate transmit and receive diversity techniques in WLANs, and introduce a number of supporting mechanisms mentioned above to increase its efficiency. A salient feature of our design is that it works above the physical layer. This allows the system to be implemented on existing commodity hardware.

2. **Frame combining:** We study the bit-error characteristics of wireless transmissions and developed an algorithm for block-based frame combining. The algorithm takes advantage of the common observed behavior in wireless communication—that errors tend to occur in bursts—in order to reduce the complexity and overhead of accomplishing its tasks.

3. **Fine-grained path selection:** We study the frame loss characteristics of wireless transmissions and developed a heuristic for fine-grained path selection. Like frame combining, the algorithm takes advantage of burst frame loss patterns in order to reduce its overhead and complexity.

4. **Evaluation:** We conducted a wide range of experiments to evaluate MRD on a 802.11-based testbed. We validated our hypothesis that path diversity improves loss resilience in WLANs, showing throughput gains up to 3× over the conventional system.

The remainder of this dissertation is organized as follows: Chapter 2 presents related work on various strategies of combating against bit-errors in the wireless medium. Chapters 3 and 4 begin by examining the empirical effects of receive and transmit diversity respectively. Then they describe the design, implementation of the MRD-RD and MRD-TD sub-systems, and present experimental results. Chapter 5 builds upon the previous two chapters, and evaluates the performance of the integrated MRD system with both receive and transmit diversity. We conclude in Chapter 6 with a summary of the dissertation and directions for future work.

Chapter 2

Related Work

Our work addresses the problem of improving the efficiency of delivering packets in wireless networks. This chapter provides a survey of other techniques that were designed to address the same problem and describes how each is related to MRD.

2.1 Error Control

One of the simplest error control methods is Automatic-Repeat-reQuest (ARQ), which applies error-detection codes to every frame and retransmits frames when bit-errors are detected at the immediate receiver terminal. To facilitate retransmissions, most WLAN link layers use a stop-and-wait protocol, which blocks transmission until the transmitter receives an ACK control frame from the receiver to acknowledge the successful receipt of a transmission or until a timeout occurs [59]. When a timeout happens, the transmitter retransmits the previous frame and waits for an ACK. The process repeats until the transmitter receives an ACK for the transmission or the transmitter reaches a retransmission limit and drops the packet. The ACK control frame in the stop-and-wait protocol is *synchronous* because the time for its transmission is always reserved, occurring soon after the transmission of the data frame.

MRD also uses *synchronous* ACKs from the link layer but does not retransmit the forward frame immediately if the corresponding link layer ACK is missing. Instead, MRD extends error control beyond the immediate receiver terminal, and sends a request-for-acknowledgment to solicit reception reports from a central controller used to collect multiple versions of the same transmission from multiple radios or APs in the network. A MRD sender then decides whether to retransmit a given frame based on the final reception status observed at the central controller, after the transmission gets a chance to recover from frame combining.

Error-correction codes are a different class of techniques. In forward error correction (FEC) [61], the sender inserts extra bits in the data stream to help the receiver recover the original data in the presence of a limited number of bit corruptions. Hybrid ARQ is an extension of this technique, which combines FEC and retransmissions to recover unsuccessful transmissions [59, 33]. Typically, FEC schemes are expressed by a coding rate k/n, which encodes k bits ($k \leqslant n$) of data with an n-bit codeword. A lower coding rate usually allows FEC to correct a greater number of corrupt bits because the number of redundant bits, n-k, will be larger.

MRD uses frame combining, which performs error correction using only the received data bits without requiring extra information from the physical layer or encoding extra bits from the sender. The idea is first proposed in [83] and then further analyzed in [30, 35]. To the best of our knowledge, we are the first to develop, implement and experimentally evaluate a block-based scheme for practical frame combining in wireless LANs. Roughly concurrent with our work, the authors in [38] develop a scheme called Simple Packet Combining for sensor networks and provide experimental results conducted on a low-speed radio testbed based on the TinyOS/Crossbow Mica2dot [5] platform.

There are various other forms of combining such as Chase combining [32] and collaborative decoding [19]. These *soft-combining* schemes are generally complex and require the receivers to exchange soft decision estimates of each data symbol, which is not accessible from any commercially available wireless device today.

2.2 Bit-rate Adaptation

Digital modulation is a physical layer process that maps n data bits into one of 2^n symbols, where each symbol corresponds to a distinct analog waveform that the sender transmits over the wireless medium. The received signal is usually some distorted version of the transmitted waveform because of attenuation, multi-path, noise and interference. During demodulation, the receiver decides which of the finite number of waveform (symbols) provides the closest match with the received signal and recovers the original data bits by reversing the bits-to-symbols mapping. Bit errors arise if the received signal is matched to an incorrect symbol.

The 802.11 physical layer transmits at a constant symbol rate but provides a set of different bit-rates. In 802.11b, there are four different bit-rates: 1 Mbit/s, 2 Mbits/s, and 5.5 Mbits/s, and 11 Mbits/s. In 802.11a, there are eight: 6 Mbits/s, 9 Mbits/s, 12 Mbits/s, 18 Mbits/s, 24 Mbits/s, 36 Mbits/s, 48 Mbits/s, and 54 Mbits/s. The bit-rates are defined by how many bits are mapped into each symbol. The 802.11a/g physical layers use FEC so their bit-rates also depends on the FEC coding rates defined for each level. In general, a high bit-rate requires mapping more bits per symbol and/or a higher FEC coding rate. Mapping more bits per symbol requires the receiver to match the received signals to a larger symbol set and increases the probability of error in presence of distortion. A higher coding rate weakens FEC's ability to correct corrupt bits. Thus, there is an inherent tradeoff between bit-rates and error rates. It is inefficient to always use a low bit-rate. Not only does throughput drop for a given link, the sender also consumes more air time in the wireless medium to transmit a frame of a given size and effectively reduce the throughput for all other wireless terminals that share the same medium [87].

There is a large body of work devoted to the development of bit-rate selection (or rate adaptation) algorithms. The goal for these algorithms is to select the highest sustainable bit-rate for a given channel condition. Most algorithms use loss [51, 55] (absence of link layer ACKs) or active sampling [24] to estimate channel conditions, while others select bit-rates on measured signal strength at the receiver [44, 76, 80]. The large channel variations that are commonly found in the wireless medium pose a significant challenge in developing efficient and practical adaptive algorithms. Current algorithms still incur significant overhead and do not work well in many environment where high variations exist, especially in the case when nodes are moving. None of the existing algorithms work unchanged in the MRD system, where multiple receivers exists and transmission terminals can change on a frame-by-frame basis. We successfully adapted one of the algorithms (implemented in [3]) to work in MRD. The modified algorithm helped MRD produce up to 3× throughput improvement, despite the high channel variations within the individual links between a given pair of transmit and receive radios.

2.3 Other Link layer Error Control Techniques

Some measurement studies have found that frame loss rates have some correlation with the size of the frame. While reducing frames can effectively reduce loss rates, especially in the presence of interference from external sources such as a microwave ovens [90], it also reduces throughput because the overhead of the header becomes proportionately larger as the frame becomes smaller. Algorithms that attempt to balance this tradeoff are presented in [69, 64, 65].

The authors of [49] presents interesting techniques to correct errors contained within link layer headers. The technique takes advantage of the observation that header contents are highly structured and change infrequently. As a result, a receiver can correct errors based on consistency checks and on Hamming distance tests against header information received in the past.

2.4 Micro-Diversity Techniques

Diversity transmissions and receptions are common techniques used to mitigate the effects of fading, and interference in wireless systems. Almost all WLAN devices contain more than one antenna; antenna selection is based frame loss rates. Recently, the IEEE incorporated a more advanced antenna diversity technique called Multiple-Input Multiple-Output (MIMO) [70] into the physical layer specifications of the next generation WLAN devices known as 802.11n [93]. Alongside of 802.11n are “smart antennas,” which are already being integrated into new 802.11a/b/g products [12, 11]. Smart antennas can be used to steer transmission signals in a particular direction to mitigate multi-path effects and reduce interference.

In general, this class of techniques, known as *microdiversity*, is tightly integrated with the physical layer and mostly helps in mitigating path-dependent effect localized at the transmitting or receiving terminal. In contrast, MRD is a solution that may be readily deployed today in any 802.11 system by just changing the software. Furthermore, MRD WLANs operate above the physical layer and may be used to collect data frames received by radios distributed across different access points at different locations to realize diversity gains at the macro level, which can yield significant performance improvement as shown in Section 3.6.2. Thus, we believe MRD and MIMO compliment one another: a WLAN operator can build a MRD WLAN using 802.11n hardware to exploit micro-diversity while using MRD to exploit path (macro) diversity over the wide-area.

Cellular phone networks also employ antenna diversity techniques, which exploit spatial diversity of strategically placed transmitters or antennas to mitigate the effects of multipath and shadowing [36, 78].

2.5 Macro-Diversity Techniques

Code Division Multiple Access (CDMA) cellular phone networks have long used “macro-diversity” to improve performance and to provide seamless handoff between base stations [45, 78, 94]. This idea is later applied to combine frames received from adjacent access points to improve uplink WLAN performance in the same way as MRD. The authors of [57] present simulated results based on a capture model but ignore protocol level issues such as ARQ. The contributions of [58, 29, 91] focus on simulation results and theoretical performance analysis, with [91] presenting results in the context of a WLAN based on Bluetooth [26] radios. In contrast, our contributions lie in the design of a macrodiversity system that works well in CSMA-based WLANs and in conducting a performance study of a fully implemented receiver macrodiversity system on a real testbed.

In Site Selection Transmit Diversity (SSTD) [40] which evolved into Fast Cell Selection in the High Speed Downlink Packet Access (HSDPA) architecture of the third generation cellular phone systems [6], the client continuously measures the pilot signals emitted by the surrounding base stations and signals the network to perform a soft-handoff to the base station that transmits a pilot with the highest received signal strength. The soft-handoff can happen on a frame-by-frame basis. The idea is similar to MRD-TD except that it relies on the receiver to make switching decisions. The architecture works well in cellular phone networks because medium access is synchronized by the base station. In WLANs such as 802.11b, medium access is randomized and distributed. Thus, feedback information may not be received by the distribution system in a timely fashion for an effective fine-grained switching to occur.

Macro-diversity can introduce duplicate and out-of-order receptions and increase the complexity of retransmission and in-order packet forwarding mechanisms. While we have not found any references that fully explains how the cellular network copes with retransmission and in-order packet delivery during a soft-handoff, the 3rd Generation Partnership Project (3GPP) has published a technical report that describes the complexity of the problem and suggests how the High Speed Uplink Packet Access (HSUPA) system might use two layers of ARQ to improve link reliability during a soft-handoff [2]. In their proposal, a client, after an uplink transmission, can obtain a MAC layer ACK signal from multiple base stations on orthogonally time-scheduled or coded channels. The client performs a MAC layer retransmission if none of the base stations respond with an ACK. The retransmitted frames are soft-combined within each base-station with any previous erroneous reception of the same frame. If a frame's retransmission limit has been exhausted at the MAC layer, the Radio Link Control protocol can optionally retransmit the frame. Successful receptions are forwarded to a Radio Network Controller, which implements a ordering buffer to reject duplicates generated by different base-stations and to ensure in-order packet delivery.

In contrast, MRD cannot rely on multiple WLAN access points to provide simultaneous ACK signaling because WLAN MAC protocols do not allow arbitrary scheduling of ACK transmissions. While WLANs do provide multiple channels using orthogonal radio frequencies, WLAN radios are designed to be simple and can be tuned to listen on only one channel at a time. Thus, to coordinate retransmissions among multiple receivers, MRD uses a request-for-acknowledgment protocol that operates above the link layer. MRD also combines frames received by multiple access points. Thus, successful error recovery is possible without retransmission in MRD.

2.6 Multi-user Diversity

Multi-user diversity [53] and medium-access diversity [48] also exploit the fact that losses at different receivers occur independently. In both techniques, a sender has a queue of packets destined for different receivers (clients) and attempts to improve network performance by scheduling transmissions to the receiver that has the best channel condition in a given moment. In contrast to MRD, the technique requires explicit receiver selection and channel quality feedback from each receiver. These requirements are necessary if the receivers are not inter-connected by a high bandwidth back-channel that MRD relies upon.

2.7 Cooperative Diversity

Instead of relying on a physical antenna array to provide diversity gains, cooperative diversity uses multiple wireless terminals distributed over an area to form a virtual antenna array for communication. The theoretical foundations of cooperative and spatial diversity is explained in [56] and by [37], which recently lead to the development of practical systems based on the same idea. ExOR [25] is a multi-hop routing system that allows multiple nodes along a routing path to receive a given transmission. Thus, a transmission can opportunistically skip hops along a path if the transmission successfully reaches a distant downstream node. Using this scheme, ExOR can reduce the total number of transmissions required to deliver a packet from a given source and destination.

ExOR is similar to MRD in that it takes advantage of the broadcast nature of the wireless medium and opportunistic receptions among multiple receivers to improve performance. However, the realized benefits of each system is quite different. MRD uses multiple receivers to reduce overall losses in the network while ExOR uses multiple receivers to improve each transmission's progress toward the destination. Recent work has improved ExOR to incorporate frame combining [38] and network coding [52] techniques to improve frame delivery rate and to improve capacity by compressing transmissions from two different paths that cross at a common relay.

2.8 Multi-radio architectures

The idea of coordinating multiple radios in WLANs has recently received considerable attention. The authors in [21, 20, 31] propose to embed multiple radios on a single device to improve energy and mobility management, enhance capacity utilization, and avoid channel failures. Some companies have even begun commercializing multi-radio access points [13].

A distributed radio bridge architecture that exploits path diversity in WLANs is proposed in [57]. MRD differs from distributed radio bridges in several important ways. The architecture of distributed radio bridges assumes that all radios in the system communicate in the same frequency. No explicit handoff is required (thus, simplifying mobility) and multiple radio bridges may participate in forwarding packets between the wired and wireless medium (thus, achieving path diversity). In comparison, MRD is a hybrid of the traditional cellular architecture and the radio bridge architecture. Frequency reuse is achieved by assigning different frequencies to each primary AP's and diversity is achieved through using secondary APs. Moreover, in the radio bridge architecture, one or more radio bridges are randomly chosen for each downlink frame transmission whereas MRD selects a single transmission site based on loss history.

There have been a few proposals to increase throughput by using multiple interfaces simultaneously in cell phone networks [84, 79, 75, 77]. Each interface may be configured to a different channel, or may use a different wireless technology so that packets may be streamed through all of them in parallel. While MRD can be modified to integrate such techniques, the focus of our work aims to improve performance by improving the *efficiency* of packet delivery without consuming much extra wireless bandwidth. The source of improvement comes from reduced losses and the length of loss bursts. We have measurements that show how MRD can sometimes provide more gains than bandwidth aggregation (Section 3.7.2).

2.9 Measurement Studies

There are a large number of prior WLAN measurement studies in indoor environments. The authors of [63] study the handoff performance of 802.11b-based wireless LANs and measure delays that range from 2 to 400 ms. Other researchers have examined loss, delay, and throughput performance in various indoor settings [34, 92, 89, 39, 41, 73]. [1] is an excellent reference for many published wireless measurement results. To the best of our knowledge, we are the first to characterize the loss correlation between different transmission paths in a WLAN and to experimentally evaluate the performance gain of fine-grained path selection and frame combining in WLANs. Furthermore, our work adds to the considerable evidence in the literature that transmission losses in the wireless medium often occur in bursts. The measurements in this dissertation were conducted using a high-rate broadcast packet stream, which allows us to sample channel conditions and evaluate the effects of path diversity for 802.11b networks in considerable detail.

The design of MRD-TD is motivated by our previous measurement study that showed how fine-grained path selection can help reduce bursty losses and thus reduce one-way packet latency in indoor mobile environments [66]. The results show that interactive video applications that have low-latency requirements can significantly benefit from such techniques.

Chapter 3

MRD-RD: The Receive Diversity Sub-system

This chapter introduces the Multi-Radio Diversity system architecture and describes MRD-RD, the sub-system that uses receive diversity to improve loss resilience at the receiving end of the communication. MRD-RD uses a central controller to collect *all* individual frame receptions among multiple radios over a high speed interconnect and can forward the correct version of the transmitted packet even when it is received erroneously at each receiver. The method of combining erroneous received versions of a frame is called *frame combining.*

MRD-RD's frame combining algorithm divides each frame into blocks. For each block, the algorithm assumes that at least one of the received copies of a frame (including any possible retransmissions) contains the correct bit values for that block. The algorithm then attempts to reconstruct the correct frame by trying every version received for each block. The process succeeds if a particular block combination passes the checksum embedded in the data frame, and fails once the search exhausts all possible block choices for each block. The computational complexity of this algorithm is exponential in the number of blocks for which different versions were received, which depends on the number of blocks in each frame. In this chapter, we show how to pick the block size and evaluate its performance using theoretical analysis and real-world experiments. This approach to frame combining is reminiscent of an old, well-studied idea called "retransmissions with memory" [83, 30], where retransmissions of erroneous frames are combined with the original transmission in an attempt to recover the correct version of the data. Our contribution is to generalize this idea using a block-based technique to incorporate the spatial dimension as well.

MRD-RD can often recover a corrupt frame without requiring a retransmission from the client, but frame combining will not always succeed. MRD-RD uses a lightweight retransmission scheme running above the WLAN link layer to further improve error recovery. The sender buffers all frames that have not yet been acknowledged (or given up on), and retransmits any frame that it believes has not been successfully received at the MRD-RD combiner (after frame combining). To prevent adverse interactions caused by the retransmission schemes at two different layers, MRD turns off link layer retransmissions but retains the link layer feedback mechanism for congestion control. Because some frames can only be recovered after frame combining, MRD-RD uses a feedback protocol between the sender and the central controller that is performing the frame combining operation. This protocol is designed to have low overhead, using a *request for ACK (RFA)* technique rather than traditional acknowledgments (ACKs) or negative acknowledgments (NACKs). With RFA, the sender explicitly requests an ACK from the central controller for certain frames, and decides whether and when to retransmit frames based on this feedback.

The rest of the chapter is organized as follows: Sections 3.1 through 3.4 describes the MRD-RD architecture, the frame combining algorithm and its theoretical analysis, the RFA scheme, and modifications to a bit-rate selection algorithm implemented on standard 802.11 devices. Section 3.6 presents the results of several experiments conducted over an in-building 802.11a-based testbed at MIT's Computer Science and Artificial Intelligence Laboratory. A noteworthy aspect of MRD-RD is that it achieves significant improvements in loss rates while consuming only a small amount of additional bandwidth. Experiments that experienced a high channel variability (e.g., a mobile WLAN client that moved over a relatively small area of about three square meters) show throughput improvements of up to three times compared to contemporary 802.11a with "autorate adaptation" [3]. Most of the work in this chapter also appears in [67].

3.1 Multi-Radio Diversity Architecture - Receive Diversity

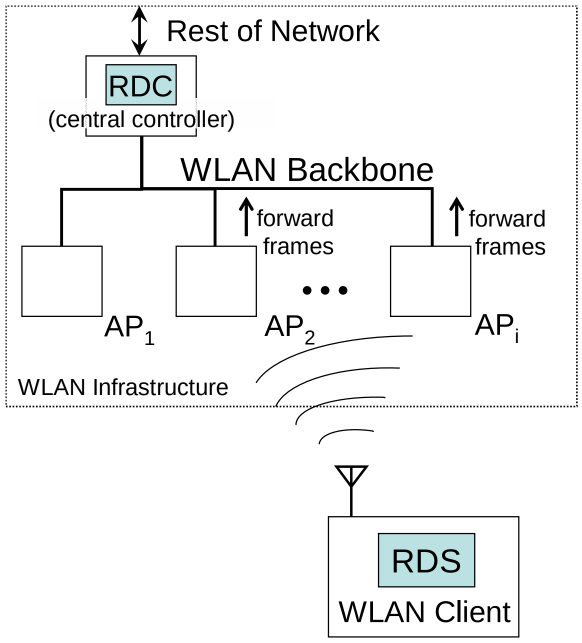


Figure 3-1: MRD-RD system architecture.

For ease of exposition, we describe the MRD-RD architecture in the context of uplink transmissions from the client to the WLAN infrastructure. The same architecture can be used when the MRD radios are co-located on the same device (either in a single AP or on the WLAN client) for downlink transmissions. We develop an alternate architecture that supports transmit diversity in Chapter 4, and an unifying architecture that integrates both transmit and receive diversity in Chapter 5;

Figure 3-1 shows the MRD-RD system architecture. Each AP in the WLAN infrastructure offers a different physical communication path between the client and the rest of the network. We configure the APs to listen on the same radio frequency so they can each receive a copy of the client's uplink transmission. The AP forwards all frames—including those that are corrupted—to a central controller called the Receive Diversity Combiner (RDC), which filters redundant data frames received by multiple radios and invokes the frame combining procedure when needed. The RDC maintains a packet buffer to deliver packets in-order to the rest of the network.

At the WLAN client sender, the Receive Diversity Sender (RDS) handles data transmissions and retransmissions. The RDS operates in between the link layer and the IP network layer. It keeps track of unacknowledged transmissions and schedules their retransmissions when it believes that the RDC has failed to receive a clean copy of the transmitted frame from any of the APs and has failed to correct their errors via frame combining. The RDS uses the request-for-ACK (RFA) protocol to obtain the results of the frame combining procedure from the RDC.

The MRD-RD architecture does not preclude cellular frequency reuse in WLANs. Frequency reuse is a common method to increase network capacity, which requires APs in neighboring cells to operate in different radio frequencies. In MRD-RD, the APs that are not explicitly associated with the client need only listen for uplink transmissions *passively*. Thus, one strategy to achieve frequency reuse is to install *passive* radios in addition to the regular, *active* radio at each AP.[1] The client associates with the active radio at each AP, which serves the regular function of transmitting management and control frames to the WLAN client, while the passive radios are configured to listen on the neighboring APs' radio frequencies. Because the passive radios never transmit a frame, they do not create any interference in the network. If installing multiple radios on a single AP is not possible, an operator can install additional passive access points in the network. As the costs of APs continue to decline, this approach is a viable way to add path diversity (for uplink communication) in WLANs.

MRD-RD assumes that there is sufficient bandwidth in the wired backbone to handle the additional traffic generated by the passive APs. This assumption is reasonable because the number of APs within reception range of a transmitter is usually low and the speed of the wired backbone[2] is usually one or two orders of magnitude higher than the wireless link.

MRD-RD does not affect the functions of handoff and security in a WLAN. As in the conventional wireless LAN, the WLAN client would associate with and perform handoffs between different active APs. Existing WLAN security standards such as WEP [15], 802.1x [17], and WPA/802.11i [8] handle encryption/decryption and other security functions in software and are easily implemented in the RDS and the RDC, assuming that the RDC can establish a secure trust relationship with each MRD radio (AP) over the network.

3.2 Frame Combining

We now describe how MRD-RD recovers error-free versions of corrupted data frames using frame combining and analyze its performance. One approach is to run a simple linear time algorithm that attempts to correct bit errors by selecting the majority bit value between three or more frames [35]. But this approach requires at least three copies of the same transmitted frame, which may not be available (without a retransmission) if only two MRD

[1]In fact, companies have begun selling radio chipsets and "radio switches" that can process and decode transmissions from multiple channels simultaneously (see, e.g., [7, 13]).

[2]In the downlink direction, the "backbone" is the internal bus interconnecting the wireless interfaces within a client.

radios are within receiving range of the sender. Therefore, we develop and analyze a block-based frame combining scheme that can work even when only two copies are available.

Suppose two copies of the same transmitted frame of size S bits are received at two different receivers. If any of the data frames passes the link layer cyclic redundancy checksum (CRC) check, the RDC decodes that as the transmitted frame and forwards it—we term this step *soft selection.* Otherwise, the RDC runs the block-based combining algorithm to recover the packet. Block-based frame combining works by first subdividing both frames into blocks, and then reconstructing the frame by assembling the blocks selected from each received frame of the transmitted packet. The process succeeds if a block combination passes the CRC embedded in the data frame, and fails once the search exhausts all possible block combinations.

For the two-frame case, the block combining steps are:

1. The input to the algorithm is two frames, $f = \{\mathcal{A}, \mathcal{B}\}$, of size S bits each, divided into N_B blocks $X = \{X_1^f, X_2^f, ..., X_{N_B}^f\}$. Let $\Delta = |\{i|X_i^{\mathcal{A}} \oplus X_i^{\mathcal{B}} \neq 0\}|$ (i.e., the number of blocks that do not have matching bit values). All matching blocks from $\mathcal{A}$ and $\mathcal{B}$ are retained unmodified.

2. Assemble a combined frame that contains $X' = \{X_1^{f'}, X_2^{f'}, ..., X_{N_B}^{f'}\}$ blocks from either frame $\mathcal{A}$ or $\mathcal{B}$. Each iteration of this step generates a new combined frame by replacing $X_i^{f'}$ with either $X_i^{\mathcal{A}}$ or $X_i^{\mathcal{B}}$ for each i where $X_i^{\mathcal{A}} \oplus X_i^{\mathcal{B}} \neq 0$.

3. If either of the CRC value embedded in frames $\mathcal{A}$ or $\mathcal{B}$ matches the CRC value computed over X', return the combined frame containing X'. Otherwise, repeat step 2 until all possible combinations of X' have been tried. If none of the block combinations X' passes the CRC check, declare a frame combining failure.

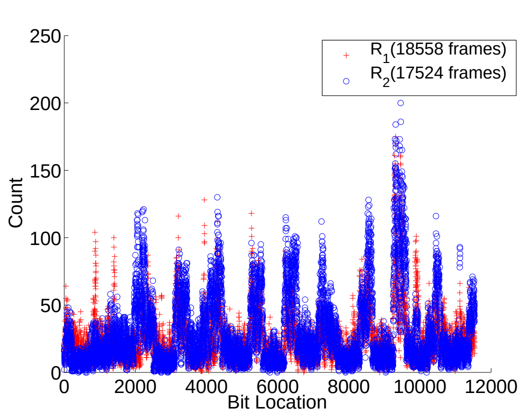
There are many ways of dividing a frame into blocks. For simplicity, we divide each frame such that blocks $X_1^f, X_2^f, ..., X_{N_B-1}^f$ contain B bits and the size of the last block $|X_{N_B}^f|$ is $\leqslant B$. Thus, $N_B = \lceil S/B \rceil$.

When the block-based frame combining algorithm declares a failure, the RDC can save the corrupt frames for possible frame combining (using either bit-majority or block-based combining) with any subsequent retransmissions of the frame. In our current implementation, the RDC saves only one of the corrupt frames and apply block-based combining to two corrupt frames at a time. Note also that the algorithm generalizes easily to more than two concurrent receptions of the same frame.

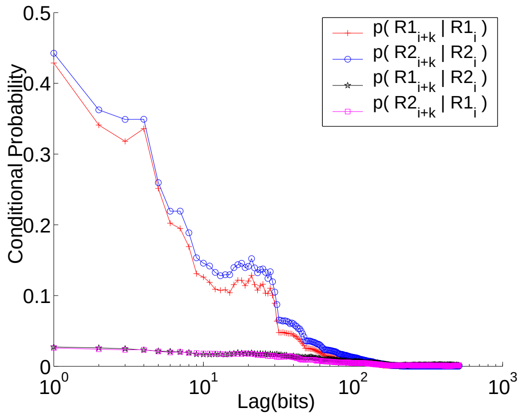
The block-based frame combining algorithm is simple but its running time is exponential in Δ, the number of differing blocks. With two copies, it needs up to 2^{Δ} CRC check operations to identify the correct combination. Since $\Delta \leq N_B$, one way to bound the number of CRC checks is to reduce N_B by increasing B. Inevitably, the frame combining failure probability will increase as the likelihood of simultaneous block errors increases with B. We analyze this trade-off next.

3.2.1 Frame Combining Failure Analysis

We analyze how the frame combining failure probability, p_f, varies with N_B under a burst bit-error channel model parameterized by a burst length b. p_f is the fraction of frames that cannot be corrected with combining out of those that could not be corrected by the soft selection in the first place. To find the overall retransmission probability, we assume that



(a) Bit-error location histogram



(b) Bit-error conditional probabilities

Figure 3-2: Figure 3-2(a) shows that the bit-errors are clustered in a regular pattern within a frame. The number in the legend indicates the number of corrupt frames received at each node. The conditional probabilities in Figure 3-2(b) suggest that bit-errors occur in bursts within a frame but bit-errors between frames received at different locations have low correlation.

each receiver observes independent losses, and multiply p_f with the independent frame loss rates (FLR) at each receiver $FLR(R_a) \times FLR(R_b)$ (i.e., the probability that the frame goes uncorrected by soft selection). We have already validated (albeit in a limited manner) the loss independence assumption in Section 1.2.1, where we showed evidence of loss independence among all pairwise combinations of simultaneous receivers in our experiment. In Section 3.6, we conduct more exhaustive experiments that show that MRD works well in practice, indirectly further validating the independence hypothesis.

Using the same experiment as in Section 1.2.1, we validate the assumption that bit-errors occur in bursts by analyzing the bit-error patterns of over 36,000 corrupt data frames for a specific pair of receivers R_1 and R_2. Figure 3-2(a) plots a histogram of the bit-error locations, which shows that the error distribution is uneven, often clustered within 100-200 bits, spaced between 800-1200 bit positions apart. At the 48 Mbits/s bit-rate, 802.11a employs QAM-64 modulation at 2/3 coding rate. This burst pattern is also observed in other node placements on our testbed and also in another 802.11b testbed deployed in an industrial environment [92].

Next, we examine how bit errors are correlated between different receivers for the same frame transmission. Let $R1_i$ and $R2_i$ represent the event that the ith bit of the transmitted frame is received in error by receivers $R1$ and $R2$ respectively. Then, $P(R1_{i+k}|R1_i)$ and $P(R2_{i+k}|R2_i)$, for $k > 0$, represents the "auto-conditional loss probability" that the $(i+k)^{th}$ bit is corrupt, given that the i^{th} bit is corrupt in the same frame. Similarly, we use $P(R2_{i+k}|R1_i)$ and $P(R1_{i+k}|R2_i)$ to represent the "cross-conditional loss probability" that the $(i+k)^{th}$ bit is corrupt, given that the i^{th} bit is corrupt in the frame received by the alternate receiver.

Figure 3-2(b) shows the auto-conditional and cross-conditional bit-error probabilities for all the corrupt frames. The cross-conditional probabilities remain flat even at the bit level. The cross-conditional bit-error probabilities for $k < 100$ are much lower than their counterpart auto-conditional probabilities, which suggests that bit-errors rarely occur simul-

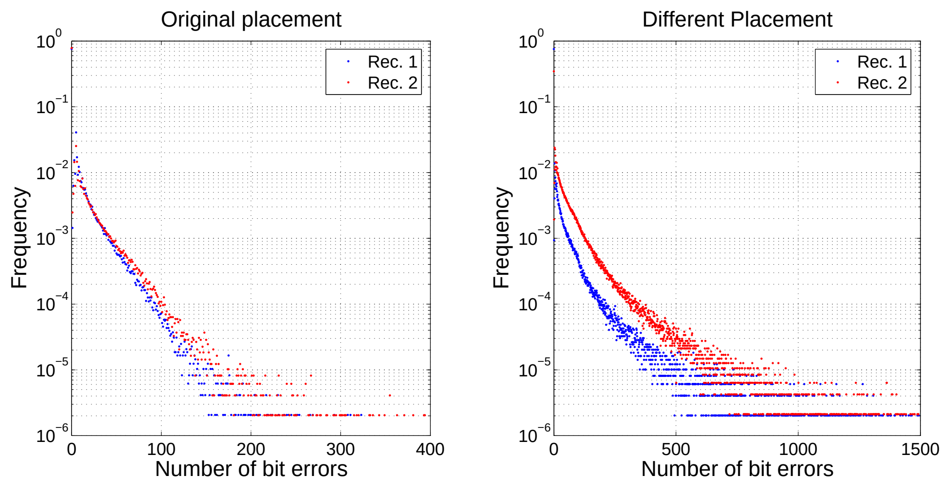


Figure 3-3: The PMF for the number of bit-errors for two different placements of the receiver pair.

taneously at nearby locations between two frames received at different physical locations. In contrast, the auto-conditional error probability at the bit level increases dramatically at small k (< 100). The increased auto-conditional probability corresponds to the burst bit-error behavior and is most likely related to the clustered bit-error patterns shown in Figure 3-2(a).

We believe that the periodic and burst nature of bit-errors observed in our experiments is due to the orthogonal frequency division multiplexing (OFDM) scheme employed in 802.11a. In this scheme, 52 separate sub-carriers are used to provide separate wireless pathways for sending the information in parallel. Four of them are used for control, and each of the remaining 48 sub-channels carries up to 1 Mbits/s summing to 48 Mbits/s. We believe that the non-uniformity of the losses is because different parts of a frame are carried by different channels, and the periodicity of bit-errors arises because the same set of data bits in each frame are consistently assigned to the same sub-channel. Indeed, QAM-64 implies that there are 6 bits/symbol on each sub-carrier and hence the bunching of $6 \times 48 \times 2/3 = 192$ data bits per OFDM symbol is consistent with this hypothesis. Also, the $800 - 1200$ bit spacing of the peaks may be caused by the time-varying nature of the channel between the stationary transmitter and receivers.

These experimental observations motivated us to develop an analytic model that allows us to examine how p_f is affected by the bit-error burstiness in the communication channel. In our model, we assume that bit-errors occur in bursts of $b \geqslant 1$ bits. Moreover, we assume that these sequences of consecutive b bit-errors are spread uniformly over the frame. Thus, if there occur d such sequences in a given frame, then it means there are a total of bd bit-errors in that frame. We neglect the effect of two individual error sequences starting within b bits of each other.

Let $D_{b,i}$ represent the number of b-bit sequences with errors in a given frame received at receiver R_i. Then,

$$\mathrm{P}\left(D_{b,i} = d | D_{b,i} > 0\right) = \eta \sum_{d'=(d-1)b+1}^{db} \mathrm{P}\left(D_i = d'\right). \tag{3.1}$$

where $\eta = (1 - \mathrm{P}(D_i = 0))^{-1}$ and $\mathrm{P}(D_i = d')$ is the probability that a frame received by R_i contains d' bit-errors.[3] We obtain the distribution of number of bit-errors empirically. Figure 3-3 shows the probability mass function of the number of bit-errors for two broadcast experiments using different node placements. We found empirically that given that a frame contains bit-errors, $\mathrm{P}(D_i = d')$ decays almost exponentially, i.e., as $e^{-\alpha d}$ where $\alpha \approx 0.01$—0.05.

In our model, we kept the average number of bit-errors per packet fixed (independent of b) and b controls only the burst size. This model of fixed sized bursts of error implies that the auto-conditional bit-error probability distribution is a step function with a jump at b. b. Even though the step function only approximates the real auto-conditional bit-error probability distribution shown in Figure 3-2(b), it encompasses certain flavors of wireless channels where losses occur in bursts.

Let us denote the set of blocks with errors at receiver R_i by $\mathcal{N}_i$. Then $|\mathcal{N}_1 \cap \mathcal{N}_2|$ represents the set of blocks that contain simultaneous errors at both R_1 and R_2.

To derive the frame combining failure probability, p_f, we make a simplifying assumption that ignores the possibility that a sequence can spread over more than one block. This assumption is reasonable when $b \ll B$, which is likely to be the case in reality.

If the sequences of bit-errors are uniformly distributed over the frame, the probability of getting at least d simultaneous block errors, conditioned on the event that receiver R_i receives a frame with d_i trains of burst errors is at most

$$\mathrm{P}\left(|\mathcal{N}_1 \cap \mathcal{N}_2| \geqslant d | D_{b,1} = d_1, D_{b,2} = d_2\right) \leqslant \frac{\binom{N_B}{d}\binom{N_B+d_1-d-1}{d_1-d}\binom{N_B+d_2-d-1}{d_2-d}}{\binom{N_B+d_1-1}{d_1}\binom{N_B+d_2-1}{d_2}}, \tag{3.2}$$

for $d < \min\{d_1, d_2, N_B\}$. The analogy with a ball placement problem is as follows. We have d_1 red and d_2 blue balls to be placed in a total of N_B bins randomly. We evaluate the probability that at least d bins contain both red and blue balls. First, we place d red balls and d blue balls in a given combination of d bins so that each bin contains exactly one red and one blue ball. Then we distribute the remaining $d_1 - d$ red and $d_2 - d$ blue balls randomly in all possible N_B bins. We end up with an upper bound because the expression counts invalid combinations that place more than $\lceil B/b \rceil$ sequences of b-bit errors within a block, which cannot happen in reality.

Because a frame combining failure occurs when $d \geqslant 1$, the conditional frame combining failure probability is simply

$$p_f(d_1, d_2) = \mathrm{P}\left(|\mathcal{N}_1 \cap \mathcal{N}_2| \geqslant 1 | D_{b,1} = d_1, D_{b,2} = d_2\right). \tag{3.3}$$

The upper bound on the unconditional probability of combining failure is

$$p_f \leqslant \sum_{d_1=1}^{\lfloor S/b \rfloor} \sum_{d_2=1}^{\lfloor S/b \rfloor} p_f(d_1, d_2) \prod_{i=1}^{2} \mathrm{P}\left(D_{b,i} = d_i | D_{b,i} > 0\right). \tag{3.4}$$

and can be computed using Expressions (3.1) to (3.4). Note that this bound is tight since the a priori probabilities for the d_i values exceeding B/b for typical values of b and B are very small. Thus in the summation in (3.4), the weights associated with the a posteriori

[3]Note that b may not divide into d' evenly so the summation in 3.1 includes all d' for a given $D_{b,i}$ such that $D_{b,i} = \lceil d'/b \rceil$.

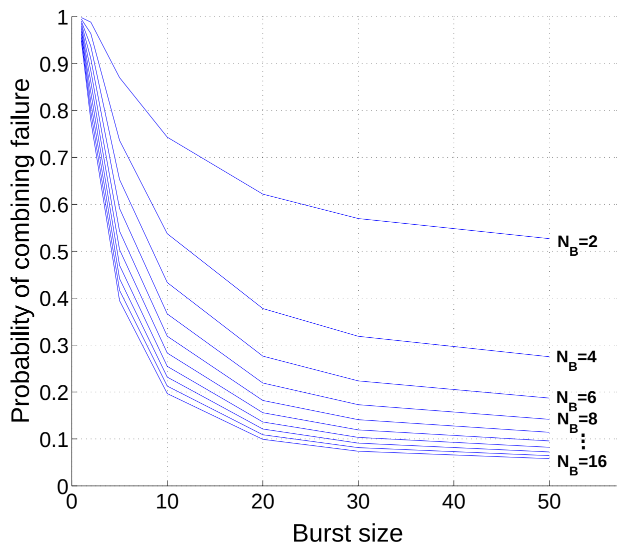


Figure 3-4: The upper bound on p_f as a function of the burst size, b. From top to bottom, the curves are plotted for the span of values of number of blocks, $N_B = 2, 4, 6, 8, 10, 12, 14$ and 16.

probabilities of invalid events of placing more than $\lceil B/b \rceil$ sequences of b-bit errors within a block are very small.

Figure 3-4 plots the upper bound on p_f as a function of the burst size b for several values of the block size, N_B. If the bit-errors are uniform ($b = 1$), p_f remains high (≈ 1) regardless of N_B. However, the auto-conditional probabilities in Figure 3-2(b) suggests that bit-errors indeed occur in bursts. In this case, we expect p_f to decrease with increasing N_B. As N_B gets larger, the difference between the two curves for a given b becomes very small, which suggests that increasing N_B beyond a certain point does not yield much improvement. Thus, we lose little performance by fixing N_B to some small value (say, 6-10) in order to bound complexity. Because p_f is a highly convex function of b, we expect the performance of frame combining to be sensitive with respect to the changes in the burstiness of the bit-errors in the channel. Moreover, the performance of frame combining will improve as the available computational power increases.

3.2.2 False Positives

We now comment on the possibility of false positives in the combining process caused by repeated trials for the CRC to check with distinct frames. In essence, the CRC is an n-bit parity check field that detects any $k < n$ bit errors and misses detection with probability 2^{-n} when $k \geqslant n$. Thus, if a 32-bit CRC is used, as in 802.11, any number of bit errors < 32 is detected. Moreover, the probability that any randomly produced frame will check the CRC is 2^{-32}, which implies that it is almost impossible for a random bit error pattern to go undetected even if a frame contains more than 31 erroneous bits.

Now, with frame combining, even though a single check leading to a false positive is highly unlikely, if we try it repeatedly many times, we may end up getting a false positive. Indeed, if the number of differing blocks in two frames is Δ, the number of swaps (and the number of tests for the CRC to check) is 2^{Δ}. For independently produced 2^{Δ} frames, the false positive probability is

$$\begin{aligned}\mathrm{P}\left(\text{false positive}\right) &= 1 - \left(1 - 2^{-32}\right)^{2^{\Delta}} \\ &\approx 1 - \exp\left(-2^{\Delta-32}\right).\end{aligned}$$

Thus, if $\mathrm{E}\left[2^{\Delta}\right]$ is close to 2^{32}, it is likely that the combining procedure leads to false positives. Even if the available computational power can perform 2^{32} CRC tests, we pick a block size that is sufficiently large (i.e., N_B is sufficiently small) so that, even in the worst case, we do not perform too many CRC checks. Hence, we guarantee by design that $2^{\Delta} \ll 2^{32}$ and keep the false positive probability sufficiently small. Our implementation uses $N_B = 6$. Furthermore, most higher layer protocols such as UDP and TCP uses an extra 32-bit CRC to provide an end-to-end data integrity check. Thus, the probability of forwarding to the application an erroneous packet with a false CRC is minuscule in practice.

3.3 Retransmissions with RFA

MRD-RD disables link layer retransmissions to allow the RDC to recover packets that the active radios receive in error. The RDS retransmits frames that the RDC fails to recover with soft selection or frame combining (i.e., a frame recovery failure). To facilitate these retransmissions, the RDS uses the *request-for-acknowledgment* (RFA) protocol to obtain the status (success or failure) of each frame transmission.

3.3.1 Design of RFA

RFA operates in between the link layer and the network layer, but uses the link layer synchronous ACKs that is implemented in most WLANs such as 802.11. A synchronous ACK is a link layer control packet that is sent by the active radio (see Section 3.1) immediately after it successfully receives a data frame. After each frame transmission, the RDS checks the link layer transmission status. A success implies that the active radio has received the transmission correctly, so the RDS can proceed to transmit the next available packet. A failure implies either a corrupt link layer ACK or a corrupt data transmission. In the former case, the RDC simply forwards the correctly received data packet or buffers it in the ordering buffer (explained below in Section 3.3.3). In the latter case, the RDC may recover the frame loss using soft selection or frame combining. If the recovery is unsuccessful, the RDC saves the corrupt frames for possible frame combining with any subsequent retransmissions of the frame.

In either case, the RDC always knows the final status of each frame transmission. Thus, when the RDS fails to receive a link layer ACK, it issues a "request-for-ACK" frame to the RDC to obtain an *MRD acknowledgment* (MRD-ACK), which contains the authoritative status of the transmission. The RDS needs to explicitly issue an RFA because only the RDS knows which frames are ACKed by the link layer.[4] To save overhead, the RDS signals a RFA by setting a flag in the frame header of subsequent data transmissions. We explain the implementation details of RFA in Section 3.5.2.

The RDS buffers the frame that fails to receive a link layer ACK for later retransmission and schedules the next available frame for transmission. The subsequent transmissions keep the wireless channel utilized while the RDS waits for the frame recovery results from the

[4]In our implementation, the MRDC also uses the RFA signal to help it flush the ordering buffer.

RDC, which can take many milliseconds. To limit the size of the retransmission buffer, the RDS may transmit up to N different frames from the first unacknowledged one. The RDS removes a frame from the retransmission buffer after K unsuccessful retransmission attempts. The RDS schedules a retransmission if the MRD-ACK indicates a frame recovery failure. If the RDS never receives an ACK from the RDC, the RDS will schedule all outstanding unacknowledged packets for retransmission after a timeout, T_s. Our current implementation uses a static timeout value of 90 ms.

There are two reasons why we chose to use the link layer ACK, instead of eliminating it and letting RDS and RDC handle retransmissions using a standard automatic repeat request (ARQ) protocol that operates strictly above the link layer. First, the synchronous ACKs are necessary for carrier-sense multiple access (CSMA) to operate properly. CSMA uses a randomized backoff window and relies on the absence of the synchronous ACK packet to detect contention and adjust the backoff window *after each frame transmission*. Because we allow transmissions to continue while the RDS waits for an MRD-ACK from the RDC, it is important to preserve the underlying CSMA channel access mechanism.[5]

Second, the wireless medium is already reserved for the transmission of synchronous ACKs in 802.11. They are designed (by means of a smaller data-to-ACK frame spacing time) to not collide with data transmissions from another nearby source. In contrast, the acknowledgments from the RDC are asynchronous and must therefore contend for the channel and suffer potential collisions. Thus, it is a good idea to avoid sending asynchronous ACKs as much as possible, especially during times when the channel quality is good and link layer losses are low.

3.3.2 Delaying Acknowledgments

To reduce the number of MRD-ACKs sent to the sender, the RDC delays the return of an MRD-ACK frame by up to D frame-transmission times, where $0 < D < N$. D should be greater than 0 because the RDC needs time to gather corrupt frame copies from the MRD radios and perform frame combining. A smaller D value would cause the system to incur higher overhead as the RDC would send MRD-ACKs more often. A higher D reduces overhead, but can cause larger transmission delay when the frame requires retransmission. In practice, the added delay is of little concern to higher layer transport protocols and most multimedia applications because D is usually set to a few frame transmission times, on the order of a few milliseconds. If $D > 1$, the RDC could process multiple frames before returning an MRD-ACK to the RDS. We expand the MRD-ACK packet with a bit-vector to indicate the final status of several packets at once, instead of spreading the acknowledgment across several different MRD-ACK frames.

3.3.3 In-order packet delivery

The RDC maintains a ordering buffer to ensure that packets are forwarded in-order to the rest of the network. When a frame requires retransmission, the RDC inserts all subsequently transmitted frames into the ordering buffer until the missing frame has been successfully recovered or has been given up on.

[5]It is conceivable to use some other channel access schemes besides CSMA (e.g., time division multiple access (TDMA)). Doing that would require introducing a major modification to the medium access control (MAC) layer of 802.11.

There are many applications, such as audio and video streaming, which are sensitive to packet delays but do not require in-order packet delivery. To cater to these applications, we can mark specific frames for out-of-order delivery. Such frames can avoid being delayed inside the ordering buffer. The implementation used to carry our experiments in this chapter does not include this feature; it has been incorporated in our most-current implementation presented in Chapter 5 (see Section 5.2.4).

3.4 Rate Adaptation in MRD-RD

Rate adaptation (or "autorate") works well when the communication channel severely deteriorates and should be used in MRD-RD when soft selection and frame combining can no longer recover frame losses effectively. Traditional autorate algorithms try to maximize throughput by using loss or signal strength information observed by a single receiver. Current autorate algorithms behave sub-optimally under MRD-RD because they do not use information observed at *all* of the diversity radios that are within range of the sender.

The interaction between rate adaptation and MRD-RD error control is an interesting open topic. Here, we present some simple modifications to an existing rate adaptation algorithm. Although these modifications may not necessarily yield an optimal algorithm for MRD-RD, we found them to work well in our experiments.

Our testbed implementation is based on 802.11 interfaces that use the Atheros 5212 chipset, which are driven by the Multiband Atheros Driver[6] for WiFi (MADWiFi) [3]. The MADWiFi driver implements an autorate algorithm that adjusts bit-rates based on the observed link layer frame loss rate. Due to the popularity of MADWiFi, the MADWiFi autorate algorithm is becoming a *de facto* benchmark. Its performance has been studied extensively in [24] and [55] and is shown to outperform the Auto Rate Fallback (ARF) algorithm that is implemented in many 802.11 interfaces on the market. We use the MADWiFi autorate algorithm as the basis of discussion, but the general ideas in this section can be applied to many other loss-based autorate algorithms.

Figure 3-5 provides a pseudo-code of the MADWiFi autorate algorithm. In our notation, *bitrate* is an integer with a range [0..MAX_BITRATE], which represents the set of discrete bit-rates available to the sender. There eight discrete bit-rates in 802.11a (6, 9, 12, 18, 24, 36, 48, 54 Mbits/s).

The MADWiFi algorithm starts by calling INIT() and invokes TXCALLBACK() to update the *numtx* and *numtxok* counters after each frame transmission. The algorithm invokes RATEADJUST() once every T seconds. If the frame delivery rate is above 90% for at least S number of successive periods, increase the bit-rate. If it falls below a minimum delivery threshold $\mathcal{D}$, decrease the bit-rate.

The original algorithm adjusts the *numtxok* counter based on link layer feedback. This approach can lead to an understated *numtxok* value in MRD-RD because the RDC can recover many frame transmissions using soft selection or frame combining. To fix this problem, we add the routine listed in Figure 3-6 to the MADWiFi autorate algorithm.

The RDS invokes MRDCALLBACK() whenever it receives an MRD-ACK. *numacked* is the number of frames (unacknowledged by the link layer) reported in the MRD-ACK that have a successful delivery status at the RDC and is added to *numtxok*. Thus, MRDCALLBACK helps the autorate algorithm maintain a correct estimate for *numtxok* as long as it receives some MRD-ACKs. Even if an MRD-ACK frame is dropped for some reason, the

[6]pci: v.0.8.6.1, hal: v.0.9.12.5, wlan: v.0.7.3.2.

```
INIT()
    $stable \leftarrow 0$
    $numtx \leftarrow 0$
    $numtxok \leftarrow 0$

TXCALLBACK()
    $numtx \leftarrow numtx + 1$
    if ($txsuccess$)
        $numtxok \leftarrow numtxok + 1$

RATEADJUST()
    if (($numtx > 0$ and $numtxok == 0$) or
        ($numtx \geqslant 10$ and $numtxok/numtx < \mathcal{D}$))
        if ($bitrate > 0$)
            $bitrate \leftarrow bitrate - 1$
            INIT()
    elseif ($numtx \geqslant 10$ and $numtxok/numtx > 0.90$)
        $stable \leftarrow stable + 1$
        if ($stable \geqslant S$ and $bitrate <$ MAX_BITRATE)
            $bitrate \leftarrow bitrate + 1$
            INIT()
    else
        $stable \leftarrow stable + 1$
```

Figure 3-5: Pseudo-code of the MADWiFi autorate algorithm.

```
MRDCALLBACK()
    $numtxok \leftarrow$
        $numtxok + \min(numacked, numtx - numtxok)$
```

Figure 3-6: A procedure that helps autorate maintain a better estimate of *numtxok* in MRD-RD.

numtxok can still be adjusted to the correct value by the subsequent MRD-ACKs because the MRD-ACKs cumulate the ACK bit vector for any unacknowledged packet. But because *numtxok* can be adjusted only upon receiving an MRD-ACK packet, long delays between MRD-ACK receptions can still cause an understatement in the *numtxok* value. This is not usually a problem in practice because 1) MRD-ACKs are always transmitted at the lowest (most robust) bit-rate to minimize loss, and 2) we set a low delay threshold (16 ms in our implementation) for transmitting MRD-ACKs.

Another problem with the original MADWiFi algorithm is that the default minimum delivery threshold $\mathcal{D}$ is fixed at 50%, which, as noted in [24], is inefficient for maximizing throughput in 802.11a/g. Let $\mathcal{D}_r$ and R_r be the expected delivery rate and effective

r	0	1	2	3	4	5	6	7
Bit-rate (Mbits/s)	6	9	12	18	24	36	48	56
$\mathcal{D}_r$	-	59	77	71	79	74	82	93

Table 3.1: The break-even frame delivery rates between bit-rate r and $r-1$ for transmitting 1500 B frames.

Scheme	Mean (Mbits/s)	Median (Mbits/s)
Slow R1	4.95	4.68
Fast R1	8.25	7.07
Slow MRD-R1	19.29	19.85
Fast MRD-R1	18.76	19.06

Table 3.2: The mean and median throughput of one second non-overlapping window samples across all five trials in each experiment.

throughput[7] using bit-rate r. Then, the throughput achieved by the lower bit-rate is the same as the current bit-rate if $\mathcal{D}_{r-1} \times R_{r-1} = \mathcal{D}_r \times R_r$.

R_{r-1} and R_r are known values and in general, $\mathcal{D}_{r-1} > \mathcal{D}_r$ because lower bit-rates are more robust against loss. To minimize loss, we set $\mathcal{D}_{r-1} = 1$. Thus, the ideal minimum delivery threshold for bit-rate r is $\mathcal{D}_r = R_{r-1}/R_r$, the ratio of the effective throughput at the lower and higher rates.[8]

In 802.11a, the typical value for R_{r-1}/R_r varies from 0.6 to 0.8. Thus, fixing $\mathcal{D} = 0.5$ is too low and causes the transmitter to maintain the current bit-rate even though its delivery rate is well below the break even point. We modified the MADWiFi algorithm to lower bit-rates according to the proper break-even ratios shown in Table 3.1.

Finally, the default values for T and S ($T = 1$ second and $S = 10$) cause the MADWiFi algorithm react too slowly to rapid changes in the channel. Instead, we set lower values $T = 0.50$ and $S = 2$ to improve its responsiveness. We ran an experiment with a high channel variability (by using mobile transmitter, described in Section 3.6.1) to compare the performance of the algorithm using different parameter values. Table 3.2 shows that the modified parameter values (Fast) helped increase throughput by about 67% over the default parameter values (Slow) for the single radio experiments using R1.

Intriguingly, the performance difference between Slow MRD and Fast MRD is negligible, suggesting that MRD-RD is relatively insensitive to the particular parameter values chosen for rate adaptation. Being able to perform consistently under different parameter values is useful, because determining the optimal parameter values for any kind of adaptive algorithm is often difficult in practice.

[7]The effective throughput is lower than the bit-rate because of link layer overhead.

[8]The actual ratios are computed using the transmission time required to transmit a frame of a given size. The transmission time needs to account for the overhead from the preamble, the link layer ACK frames, and inter-frame delays.

3.5 Implementation

This section describes the implementation of the MRD-RD system and the RFA protocol in detail.

3.5.1 System Implementation

We implemented the MRD-RD system using commodity contemporary Pentium class PCs running Linux Kernel 2.4.20 and 802.11a/b/g wireless interfaces (Netgear WAG311 PCI-card on the PC and Proxim 8480 PC-Card on the laptop) based on the Atheros 5212 chipset. We modified the MADWiFi driver to implement the RDS component for 802.11a/b/g WLAN clients.

As described before, the primary function of the RDS is to schedule retransmissions. To handle retransmissions within the driver software, we disable the wireless interface from retransmitting packets by setting the retry limit to zero. During our experimental evaluation, we discovered that doing so caused the distribution of frame inter-transmission times to peak at the nominal packet transmission time, despite many transmission losses. In other words, setting a zero retry limit also disabled exponential backoff in the 802.11 interface. It turns out this is the behavior mandated by the original 802.11 standard [15]: the contention window should reset to the lowest value after a packet reaches its retransmission limit.

Consequently, our current MRD-RD implementation does not include CSMA exponential backoff. However, future releases [9] of the MADWiFi driver [3] will include software support for 802.11e [46], which includes a software API to allow the driver to modify the contention window size. Meanwhile, we have disabled exponential backoff in *all* of our experiments to make fair performance comparison between the 802.11 standard and our MRD-enhanced 802.11 system.

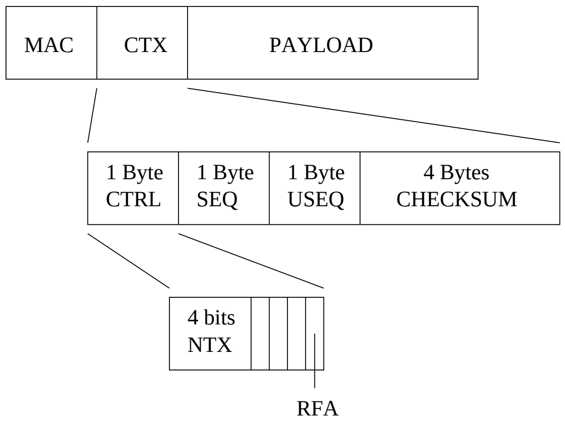
We used desktop PCs equipped with 802.11 wireless interfaces as access points. One AP acts as the active radio and is configured to run in the MADWiFi's "AP Master" mode. The passive radios are configured to run in MADWiFi's "Monitor" mode. On each of the APs, we run a user-level daemon to capture data frames from the wireless interface and forward them over a wired backbone (100 Mbps Ethernet in our experiments) to the RDC running on another PC.

For increased efficiency, the AP daemon performs the CTX header checksum (see the next section) and drops frames that cannot be used for frame combining (i.e., those frames with a corrupt header). Because the RFA protocol does not require the client to acknowledge the receipt of an MRD-ACK, the AP daemon prepends the target client's MAC address in the MRD-ACK payload and transmits each MRD-ACK as a broadcast frame. Broadcasts saves the transmission of link layer ACK frames in unicast and the benefit is much larger than the cost of expanding the size of the MRD-ACK payload. In our actual implementation, the AP daemon writes the target client's 6-byte MAC address in the source address field of the 802.11 header, thus saving us from expanding the MRD-ACK payload at all. We transmit the MRD-ACK packet at the lowest data rate (6 Mbps for 802.11a/g and 1 Mbps for 802.11b) for robust delivery.

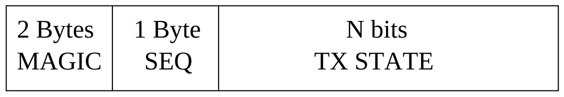
Because the CRC computation is the bottleneck of the frame combining process, it is important to make it as efficient as possible. The RDC currently implements a widely-used 8-bit table lookup algorithm to compute the 32-bit CRC checksum of a combined frame. Although the algorithm is simple, it is rather inefficient to process the entire frame to compute a new CRC value when the bit values for only a small portion of the frame changes during each iteration of the frame combining algorithm. Although not implemented, there are incremental CRC algorithms that can reduce the running time of repeated CRC checks by over an order of magnitude [27, 81]. The computation savings can be used by the frame combining algorithm to improve frame combining success rate through using smaller block sizes.

We implemented the RDC as a user-level daemon running on a 1.5 GHz Pentium 4 PC. Implementing the RDC as a user-space daemon facilities debugging and running diagnostics. It forwards clean or corrected packets to the tunneling driver so that the Linux kernel can forward the packet using `iptables`.

3.5.2 Implementation of RFA



(a) CTX Header in the transmitted data frame



(b) MRD-ACK Packet

Figure 3-7: MRD-ACK control information.

Figure 3-7(a) shows the headers used by RFA. For every data frame transmission, the RDS inserts a 7-byte Combiner Transmit (CTX) header that prepends the payload of the MAC layer frame. The CTX header contains a *ctrl* field, which uses 4 bits to indicate the number of attempted transmissions (*ntx*) for the current data frame, 1 *rfa* bit to indicate that the sender has pending unacknowledged frames and is requesting for acknowledgment, and 3 unused bits reserved for future options as discussed in Section 5.2.4. The 1-byte *seq* field labels the sequence number of the data frame, while *useq* labels the oldest transmitted data frame in the RDS buffer that has not been acknowledged by the RDC. When frame *useq* exceeds its retransmission limit, the RDS advances *useq* to the *seq* number of the next unacknowledged frame in the retransmission buffer (if any). This allows the RDC to detect frames that exhausted its retransmission limit and flush the blocked frames from the ordering buffer.

The RDC uses the source address in the MAC header and the *seq* value in the CTX header to identify the frames that belong to the same network layer packet. When the RDC receives at least 2 corrupt data frames that correspond to the same packet, it attempts frame combining on the payload part of the data frame. Since it is important that the RDC correctly identifies the frames that belong to the same packet, RFA uses a 4-byte CRC to protect the MAC and CTX header. If either the MAC or the CTX header is corrupted, the RDC drops the entire frame.

The MRD-ACK packet contains a 2-byte "magic" value that is used to distinguish the MRD-ACK packet from other downlink data payload,[9] a 1-byte sequence number, and an N-bit bit vector to indicate the success or failures of up to N consecutive frames. The sequence number is the *seq* value of the first data frame in the bit vector being acknowledged. The RDS uses the link layer data frame checksum to detect errors in the MRD-ACK packet.

The size of the MRD-ACK payload is small (25 bytes in our implementation). Thus, its overhead is dominated by the preamble and header associated with the 802.11 frame. We can potentially decrease overhead further by piggybacking MRD-ACK packets on data frames being transmitted in the same direction.

Our RFA implementation allows the RDC to delay ACK transmissions in terms of the number of successive transmissions made by the RDS. Thus, RDC can delay an ACK either by a timeout of length equal to a duration of D packet transmissions or by counting D packet transmissions from the RDS. Note that retransmitted frames are counted as a transmission while extra frames that are simultaneously received by different MRD radios should not be counted. Because both types of frames have identical *seq* values, the RDC uses the *ntx* value to distinguish the retransmitted frames.

The RDC sends MRD-ACKs to the RDS via the active radio (i.e., the AP with which the WLAN client is associated for RDS running in the WLAN clients). The RDC may also independently use fine-grained path selection (Chapter 5) to choose the most reliable diversity radio for transmitting the MRD-ACK packet to the WLAN client.

3.6 Evaluation

We conducted several experiments to evaluate the performance of MRD-RD under different conditions. We divide the presentation of the results into two categories, *HIVAR* and *LOVAR*, based on whether the WLAN client was experiencing a high or low degree of channel variability during the experiment. To create a high channel variability environment in HIVAR, we use a client transmitter that is set in motion during the experiment, while we use a stationary transmitter in LOVAR. We describe the general setup for HIVAR and LOVAR experiments next.

3.6.1 Setup

We chose to conduct experiments in 802.11a mode to avoid interfering traffic from the production 802.11 WLAN in our lab. In all our experiments, we configure one of the AP receivers (R1 or R2) to be an active AP running in *Master* mode. We configure the other AP receiver to run passively in *Monitor* mode. We configure the client sender C to run in 802.11 *Managed* mode. We run the RDS on the WLAN client to evaluate the performance for upstream traffic.

In all of the experiments, we set a maximum retransmission limit of 7 (one initial transmission plus up to seven retransmissions). The MRD-RD experiments used a MRD-

[9]Instead of using "magic", we should label the MRD-ACK with a unique value in the Ethernet `type` field [85]. We used the magic value in our implementation to facilitate logging using standard tools like `tcpdump` [4] during our experiments.

ACK delay of $D = 8$ packet transmissions, a sender buffer size of $N = 64$ packets, and a retransmission timeout of $T_s = 90$ ms. We pick $B = 256$ bytes ($\therefore N_B = 6$), such that the maximum processing time to search through 2^{N_B} block combinations is less than S/r, where S is the transmitted frame size and r is the bit-rate. Bounding B in this way helps prevent the processing queue at the RDC from building up.

In each experiment, the WLAN client sends 100,000 1472-byte UDP packets as fast as possible to saturate the wireless channel. We repeat each experiment for five trials. To help ensure each trial begins with fresh states, we re-load the wireless interface driver in between trials. On the first transmission of each packet, we insert a timestamp into the frame's payload. The timestamp remains unchanged on frame retransmissions. The timestamp allows us to measure and compare the packet delivery delay between MRD-RD and the communication schemes that use single-radio terminals (single-radio). Also, the payload of the packet contains a known bit pattern so that we can post-process the trace to analyze the probability of frame combining failure p_f as a function of different block sizes B.

Each MRD-RD experiment involves two sub-experiments: in the first set (MRD-R1), we configure R1 to be the active AP with which the client associates and R2 to be the passive AP. In the second set (MRD-R2), R2 is active and associates with the client. We compared the performance using different active APs because the RDS schedules retransmissions based on the link layer feedback from the active AP.

As mentioned in Section 3.5, performing software-based retransmissions in the driver effectively disables exponential backoff in the wireless interfaces' firmware. To make a fair performance comparison between communication schemes, we used software-based retransmissions (and thus, disabling exponential backoff) in all of our experiments, including the single-radio schemes. We discuss how disabling exponential backoff might affect our evaluation results in Section 3.7.

Because wireless communication is sensitive to the physical environment, we do not claim that the results of the experiments presented here are exhaustive and representative of *all* situations. Our main objectives are to conduct a set of experiments to illustrate the performance gains that MRD-RD can achieve in the implemented system under a *real* environment with different degrees of channel variability, and to analyze the properties of the MRD-RD system in depth.

We present the results of our HIVAR and LOVAR experiments below. The HIVAR experiments used the modified autorate algorithm as described in Section 3.4, but the LOVAR experiments were conducted before we implemented the modifications. Thus, the LOVAR experiments use the standard MADWiFi autorate algorithm, which could have reduced the performance of MRD-RD for those experiments.

3.6.2 High Channel Variability (HIVAR) Experiments

We compare the performance of single-radio schemes against MRD-RD when the client experiences a high degree of channel variability. Figure 3-8 illustrates the location of our APs and client in our HIVAR experiments.

Throughput

We define throughput to be the sum of the payload bits from unique frames received divided by the time elapsed between the first and last frame receptions. Note that the throughput

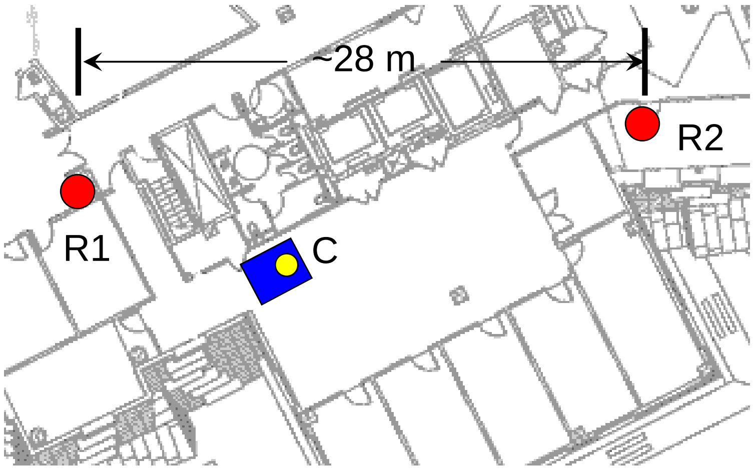
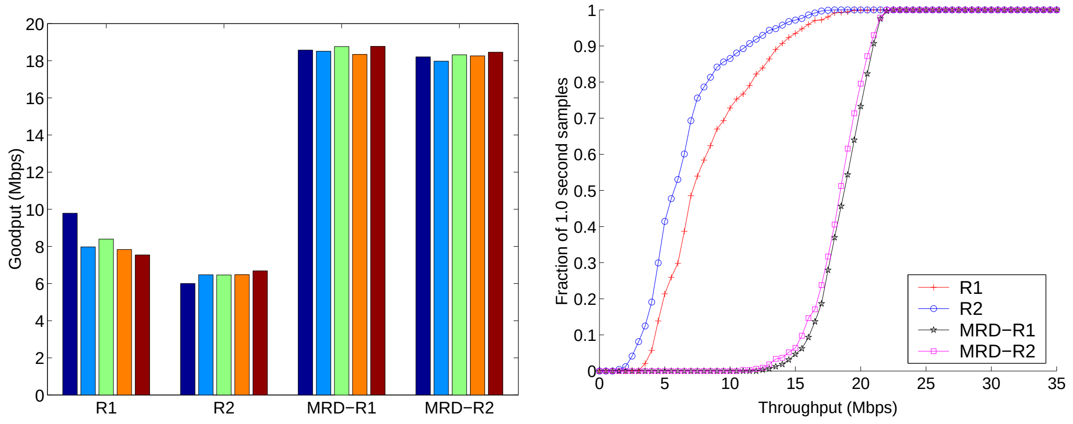


Figure 3-8: Setup for the HIVAR experiments. R1 and R2 are stationary receivers. C is a laptop transmitter client that was carried by a walking person who covered a 1.5 m × 2 m area during the experiments.

metric accounts for the overhead of the CTX header, MRD-ACK transmissions, and all the processing delay associated with MRD-RD.



(a) Throughput (b) One-second throughput distribution

Figure 3-9: HIVAR Throughput Analysis. Left: Throughput of single-radio and MRD-RD experiments; throughput of each trial is represented by a bar within an experiment set. Right: Distribution of throughput averaged over non-overlapping one-second window samples.

Figure 3-9(a) shows the goodput of each trial of our experiment. The average throughput over five trials for the single-radio experiments R1 and R2 were 8.25 Mbits/s and 6.42 Mbits/s, which are far below 802.11a's theoretical maximum UDP throughput of 31 Mbits/s. The high channel variability caused by mobility and distance in our HIVAR experiments has significantly reduced throughput for the single-radio experiments. Despite the harsh channel conditions, MRD-R1 and MRD-R2 maintained an average throughput of 18.7 Mbits/s and 18.36 Mbits/s respectively, which constitute improvements of 2.27× and 2.23× over R1 (and even more over R2). The gains of MRD indirectly confirms that path diversity exists in our experiment.

Moreover, the throughput of both MRD experiments are higher than the sum of the throughput values from both single-radio experiments. Hence, the results suggest that MRD could achieve a higher throughput than a scheme that aggregated the bandwidth of two radios to transmit data over orthogonal radio frequencies. Another words, our results show that MRD *can* outperform a system that uses roughly twice the wireless bandwidth as MRD!

We plot the throughput distribution of the one-second non-overlapping window samples in Figure 3-9(b). For R1 and R2, 80% of the samples are between 4-10 Mbits/s and fewer than 10% of the samples achieved a throughput more than 15 Mbits/s. In contrast, MRD-RD achieves a throughput greater than 15 Mbits/s for more than 85% of the samples. These results suggest that *even* if we allow the WLAN client for the non-MRD schemes to perform handoffs every second, the average throughput will remain well below 15 Mbits/s.

Both MRD-R1 and MRD-R2 achieved similar throughput results. This suggests that the performance of MRD-RD is relatively insensitive to the choice of active AP, even when there is a significant difference in link quality between the two APs.

Source of Improvement

The large throughput improvement comes from the reduction in frame loss rate achieved by MRD-RD. Table 3-10(a) summarizes the statistics of the raw frame loss rate (FLR) observed at the active AP in each sub-experiment and the ratio of the lost frames that were recovered (frame recovery rate, FRR) by MRD-RD. The active APs in both sub-experiments suffered a raw FLR of about 35% and 39% but MRD-RD was able to recover 50% and 57% of them, respectively.

The HIVAR experiments used the modified autorate algorithm as described in Section 3.4. Because MRD-RD was able to conceal a large number of losses from the rate adaptation algorithm, the sender was able to maintain a high bit-rate throughout both sub-experiments, as depicted in Figure 3-10(b), where over 90% of the frames were transmitted at a bit-rate of 24 Mbits/s or higher. In contrast, the single-radio schemes suffers a high loss rate at the high bit-rates. Consequently, these schemes operate at low bit-rates. Actually, the selected bit-rates in R1 and R2 are spread across several of the low bit-rates due to the high degree of channel variability experienced by the client.

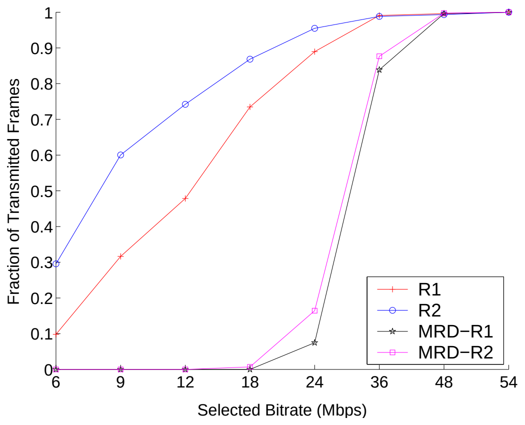
These results highlight the importance of MRDCallback(). If the procedure were not added to the autorate algorithm in MRD-RD, the link layer frame losses observed at the active AP would have been exposed to the autorate algorithm, causing RDS to operate at the same low bit-rates as $R1$ and $R2$ in Figure 3-10(b).

We decompose the recovered frames into frames recovered by soft selection (FRR_{SS}) and block-based combining (FRR_{FC}). Thus, $FRR = FRR_{SS} + FRR_{FC}$. Our results show that 85% and 90% of the gains in MRD-R1 and MRD-R2 were achieved by soft selection (i.e., those frames that were received correctly by the passive AP but not by the active one).

There are two possible explanations for the relatively small fraction of frames recovered by frame combining: i) there were few opportunities for running the packet combining either because most of the transmissions were already corrected by soft selection or because the RDC did not collect enough valid corrupt frames (due to corrupt headers, etc.) to perform the combining; or ii) there were many frame combining attempts but most of them failed to recover the correct frame.

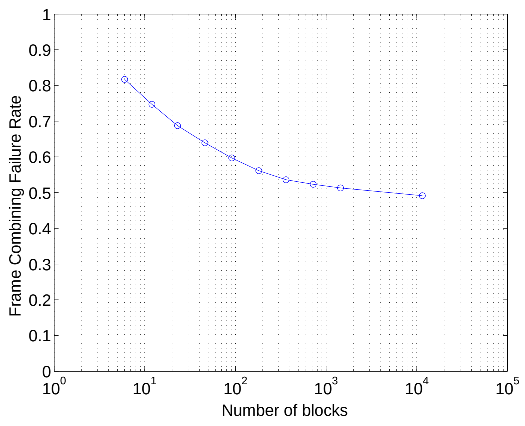
Experiment	FLR	FRR	FRR_{SS}	FRR_{FC}
MRD-R1	0.345	0.497	0.423	0.073
MRD-R2	0.391	0.573	0.515	0.058

(a) Frame loss/recovery rates.

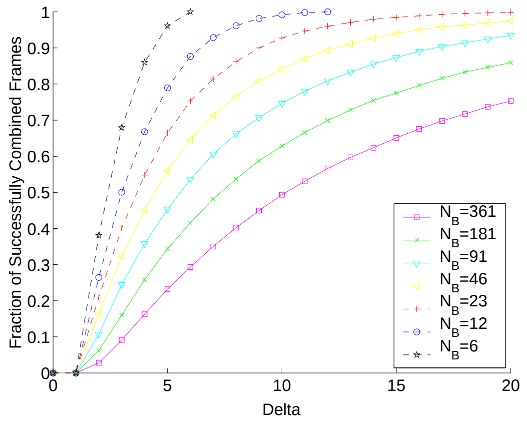


(b) CDF of selected bit-rate.

Figure 3-10: HIVAR source of improvement analysis. Top: Frame loss (FLR) and frame recovery rates (FRR) averaged over 5 trials of the high channel variability experiments. FRR is decomposed into two sources of recovery: soft selection (FRR_{SS}) and frame combining (FRR_{FC}). Bottom: Distribution of selected bit-rate for each transmission.
one-way packet delivery time.



(a) Failure rate.



(b) CDF of the number of differing blocks, Δ.

Figure 3-11: Trace-driven simulation of p_f and Δ for various values of N_B in the HIVAR MRD-R1 experiments.

We analyzed the number of successful and failed frame combining attempts. The total number of frame combining attempts was high, constituting 34% and 26% of the total number of frames that were not successfully received by the active AP in MRD-R1 and MRD-R2. Although there were many opportunities for error recovery with frame combining, about 80% of those attempts failed to correct the errors in the transmitted frame in both sub-experiments.

One cause for the high failure rate is the low number of block subdivisions in a frame in our implementation ($N_B = 6$). We post-processed the data trace of our experiments to analyze how p_f varies with other values for N_B and plot the results in Figure 3-11(a).[10] The plot shows that p_f drops as N_B increases, which is consistent with the analytic model for burst bit-error channels that we developed in Section 3.2. For example, p_f drops from 80% to 60% when $N_B = 91$ (i.e., $B = 16$ bytes).

As discussed in Section 3.2, increasing N_B can potentially increase Δ, the number of differing blocks between two frames. To avoid overloading the RDC, we may need to abort the frame combining operations for frames received with a large Δ. Thus, a high Δ for a large fraction of combining attempts can offset the performance gain from increasing N_B. Figure 3-11(b) plots the distribution of the number of unmatched blocks (Δ_{succ}) for the successfully combined frames at various N_B. For $N_B \leqslant 91$, the 75th percentile Δ_{succ} value are much smaller than N_B (e.g., for $N_B = 91$, the 75th percentile of Δ_{succ} is 10).[11] This suggests that we could improve the performance of frame combining by re-running our experiments with a larger N_B value.

Finally, the relatively low overhead of RFA allows MRD-RD to achieve high gains. The number of MRD-ACKs transmitted constitute fewer than 7.5% of the total number of transmitted packets and fewer than 0.1% of the total number of transmitted bytes. The overhead of inserting an extra 7-byte CTX header to the 1500-byte packet payload is also negligible.

Frame Losses and Bit-rates

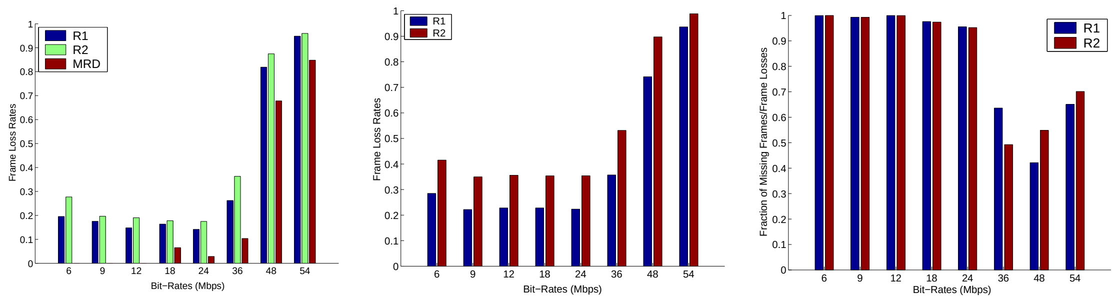
We analyze how FLR varies with bit-rates to gain further insight and to explain why the single-radio experiments performed so poorly against MRD. Our results shows that a large fraction of frame losses were frames that were never received at the individual receivers, and that the rate of which such losses occur remain the same at every transmission bit-rate. Thus, reducing the bit-rate did not improve the delivery rate or the throughput for the single-radio reception schemes.

Figure 3-12(a) shows that the average FLR of the HIVAR single-radio and MRD experiments at different bit-rates. As one might expect, the FLR decreases substantially as the bit-rate decreases from 54 Mbits/s to 36 Mbits/s in all of the experiments. The MRD experiments never had to use any bit-rates lower than 18 Mbits/s because $1-FLR$ never fell below the break-even frame delivery rates within any 500 ms window (the rate adaptation update interval) at 18 Mbits/s (Section 3.4).

In contrast, the average FLR for R1 and R2 remain high, ranging from 16% to 28%

[10] Recall that p_f excludes those frames that are successfully delivered by soft selection (Section 3.2.1). While the majority of frame combining attempts were performed for corrupt frames that were simultaneously received by the APs, a significant fraction of the frame combining attempts were performed with retransmitted frames. For simplicity, we excluded the retransmitted frames in our post processing analysis. Nonetheless, our results should remain representative because the retransmitted frames should have an independent bit error behavior similar to the simultaneously received frames.

[11] Performing $2^{\Delta} = 2^{10}$ frame combining checksum operations for a 1500-byte packet takes about four milliseconds on a 3.2 GHz Pentium IV PC. The processing time is rather large and may cause the processing queue to build up at the RDC. However, it should be possible to reduce the processing time substantially by using an incremental CRC update algorithm [27, 81] or using specialized hardware to perform the CRC calculation.



(a) (b) (c)

Figure 3-12: Left: Frame loss rates vs. bit-rates from the HIVAR experiments. Middle: Frame loss rates vs. bit-rates from the uniform sampling broadcast experiments. Right: Frame missing rates (*i.e.,* the fraction of lost frames that were never received) vs bit-rates.

even at low bit-rates between 6 and 24 Mbits/s. At first glance, the results are rather surprising because one should expect the FLRs to decrease substantially with decreasing bit-rates. However, the high FLRs at the low bit-rates could have been inflated because the experiment does not sample the frame loss rates uniformly over a given physical region for a given bit-rate. Instead, the transmitter adapted the bit-rates as it moved. So, the transmitter might have used low bit-rates only in a few, specific "bad spots" where the channel happens to be extremely poor. As a consequence of non-uniform sampling, our trace could observe a higher-than-expected FLRs at low bit-rates.

To determine whether our results are skewed by non-uniform sampling or whether the low bit-rates were ineffective in decreasing the FLR in the single-radio schemes, we repeated our HIVAR experiment, except that the transmitter in the new set of experiments transmitted consecutive broadcast frames (*i.e.,* no retransmissions) at different bit-rates, cycling among all of the eight available bit-rates in 802.11a. It takes about 8 ms to cycle the broadcast frames through all eight bit-rates. Hence, the transmissions in the new set of experiments can sample the loss rates uniformly over the transmitter's movement area for the different bit-rate. Moreover, because the transmitter rotates bit-rates among consecutive transmissions, the transmissions that used a given bit-rate experience similar channel variations to the transmissions that used another bit-rate.

Figure 3-12(b) shows the average FLRs for R1 and R2 at each bit-rate from five trials of the uniform sampling experiment. Similar to our original HIVAR experiments, the FLRs of our uniform sampling experiments remain high at low bit-rates. There are two possibilities (or a combination of them) that could lead to our observed results: 1) the channel condition varied immensely within the transmitter's movement area and that there were certain bad spots where the bit-error rates remain high even for low bit-rates and 2) there is another type of transmission error that was causing high frame loss rates, despite the low bit-error rates provided by the low bit-rates. In the former case, the receiver would receive many frames that contain at least one bit-error. In the latter case, the receiver would miss the entire transmission and would not make an upcall for a frame reception in the protocol stack.

We analyzed the frame trace of our original HIVAR experiments and counted the number of frames that were never received by the receiver. Figure 3-12(c) plots the number of missing frames as a fraction of the total number of frame losses for the single-radio experiments.

At high bit-rates, the missing frame ratio is 62% and 76% respectively. From 6 Mbits to 24 Mbit/s, the missing frame ratios reach near 100% in both single-radio experiments. The results suggest that the low bit-rates were able to reduce bit-error rates: a frame almost never suffers from any bit-errors whenever it is received. On the other hand, the results also suggest that the receiver is suffering from some other types of errors other than bit-errors within the payload. In 802.11, each transmission frame is preceded by a special training sequence of bits called the preamble that allows a receiver to detect the start of an incoming transmission and synchronized with the transmitter to decode the rest of the frame. While we cannot identify what caused the receiver to drop entire frames at low bit-rates, it is likely that the receiver had suffered from some type of preamble detection, perhaps because of the complex nature of the process [42] and/or random noise in the receiver electronics.

These results confirms the observation made in [24], in which low bit-rates may not always help reduce frame loss rates as one might expect. The algorithm we used assumes that low bit-rates reduce frame loss rates, which is not always true in practice (Section 3.6.2). As a result, the algorithm reacts to losses by lowering the bit-rate. Not only does it fail to reduce frame losses, but it also decreases throughput as transmissions consumes more channel time at lower bit-rates.[12] However, our results show that MRD is able to overcome losses and maintain high bit-rates, which suggests that MRD could use both path and radio diversity to overcome problems that cannot be easily be mitigated by rate adaptation alone.

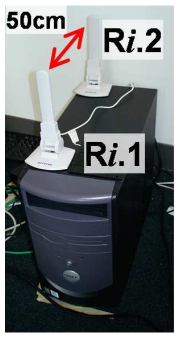
Performance of co-located radios

We measure the performance of MRD-RD for the case when the active and passive radios are co-located at the same site with each radio operating in the same radio frequency. This scenario is roughly equivalent to running MRD-RD in the downlink direction, where the MRD radios are co-located on the wireless client. We also compare our results with those obtained with MRD radios that are separated over a large distance, and show the benefits of realizing diversity gains in the wide-area.

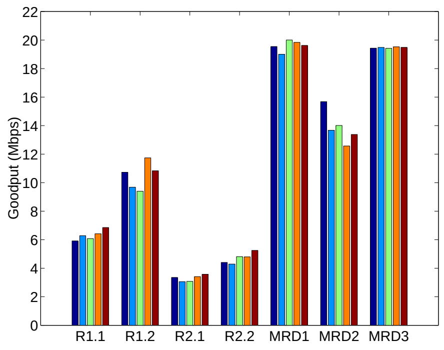
We use the same setup as in Section 3.6.1 and install one additional radio at each access point R1 and R2. The antennas of the co-located radios are placed at about 50 cm apart (Figure 3-13(a)). We repeated the HIVAR experiments for the single-radio schemes for the first radio at R1 (R1.1), and for the second radio at R1 (R1.2). Similarly, we repeated the single-radio schemes for each of the two radios at R2 (labeled respectively as R2.1 and R2.2). We ran three HIVAR MRD-RD experiments: MRD1 uses the co-located radios at R1, with R1.2 being the active radio, MRD2 uses the radios at R2, with R2.2 being active and MRD3 uses radios R1.2 and R2.2 at R1 and R2 respectively, with R1.2 being the active radio. The active radios were selected on the basis that they have the better link quality between the two radios.

Figure 3-13(b) shows the measured throughput of our experiments. Each experiment is repeated five times with the result of each trial plotted on a different color bar. R1.1

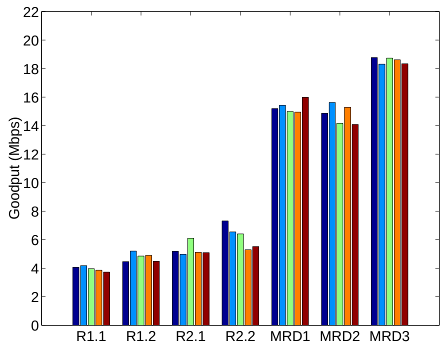
[12]SampleRate [24] is an algorithm that is designed to overcome this deficiency by sampling the frame loss rates at all the bit-rates and selecting the bit-rate that most recently maximized throughput. However, SampleRate's adaptation interval is long (10 s) and may not be appropriate for highly varying channel conditions. We advocate modifying the link layer to provide additional feedback to help the transmitter quickly distinguish between losses that can be mitigated by low bit-rates (*i.e.,* frames that contain bit-errors) and other types of losses (*e.g.,* preamble synchronization failures). To provide such feedback, the receiver can return a synchronous ACK whenever it receives a frame, with an extra bit to indicate if the frame's received payload contains any bit-errors. The absence of an ACK indicates that the receiver has dropped the entire frame. A rate adaptation algorithm can then use this feedback to select bit-rates accordingly.



(a)



(b)



(c)

Figure 3-13: Left: Co-located Radio Node Setup. Middle: Performance comparisons between co-located MRD-RD radios and widely separated MRD-RD radios. Each colored bar represents the results of one experiment trial. Right: The same performance comparison repeated at different transmitter location.

and R2.1 were the same radios that were used in the previous HIVAR experiments, but we measured a much lower average throughput of 6.3 Mbits/s and 3.3 Mbits/s for R1.1 and R2.1, than throughput of 8.25 Mbits/s and 6.42 Mbits/s measured in the previous experiments. One possible cause for the difference is that the new set of measurements are conducted five months later than the previous ones and the furniture and obstacles in the surrounding environment changed in the meantime. Interestingly, the average throughput of R1.2 is about 1.66× greater than R1.1, even though the two radios are only 50 cm apart. These observations show how unpredictable wireless performance can be in an indoor environment.

Figure 3-13(b) shows that the MRD1, MRD2, and MRD3 schemes achieve average throughput of 19.6 Mbits/s, 13.9 Mbits/s, and 19.5 Mbits/s respectively, representing an improvement between 1.88× to 2.9× over their non-MRD counterpart. Even though the MRD radios are placed in close proximity, MRD1 achieved the best result. Indeed, the correlation of signal envelopes between two closely-spaced antennas is around a quarter wavelength [78]. This corresponds to less than a few centimeters at the GHz frequencies. Thus, two radios that are separated by about 50 cm are quite uncorrelated and would exhibit high diversity gains.

MRD2 underperforms both MRD1 and MRD3, because both radios at R2 are relatively far away and receive relatively weaker signals from the laptop transmitter. Thus, MRD2 illustrates a case where co-located radios can diminish the gain of MRD-RD. When the MRD radios are co-located, they tend to share similar levels of path loss to the transmitter. Even though the instantaneous losses are uncorrelated, the average loss rates at each radio becomes increasingly correlated with distance, which eventually cancels out the diversity gains in MRD-RD.

In contrast, MRD3 achieves almost the same performance as MRD1, even though the link quality for the passive radio (R2.2) in MRD3 is much lower than the passive radio (R1.1) in MRD1. One possible explanation for this result is that the wide separation distance between the MRD radios reduces the average path loss correlation to each receiver; *i.e.,* as the mobile laptop transmitter moves from one end of the test area to another, the average received signal strength might become weaker to one of the radio but stronger for another.

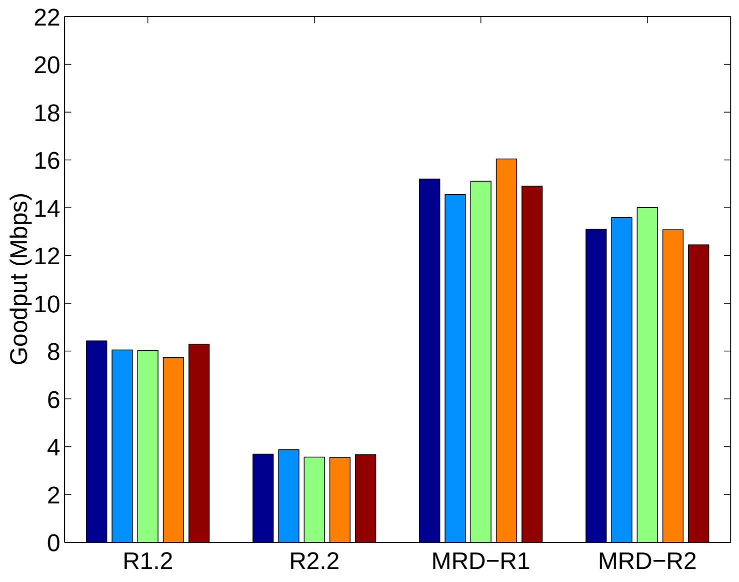


Figure 3-14: Measured TCP throughput under various non-MRD-RD and MRD-RD schemes. Each colored bar represents the results of one experiment trial.

Together, the wide-area MRD radios compliment one another and provide better overall signal reception coverage for the mobile transmitter.

To test this hypothesis, we shifted the laptop's movement area by about one meter towards R2 and repeated our HIVAR experiments. We plot the measured throughput in Figure 3-13(c). Because the average distance to R2 has reduced, we observe an increased average throughput of 5.30 Mbits/s and 6.22 Mbits/s for R2.1 and R2.2. In contrast, R1.1 and R1.2 suffered a large throughput reduction to 3.96 Mbits/s and 4.78 Mbits/s. The reduction is especially drastic for R1.2, perhaps due to the lack of line-of-sight signal propagation around the hallway between the laptop and R1 (Figure 3-8).

We observe a similar trend for the MRD1 and MRD2 experiments. Compared to the previous experiment, MRD1's throughput reduced to 15.3 Mbits/s while MRD2's increased to 14.8 Mbits/s. On the other hand, MRD3 maintains a high throughput and shows only a small throughput decrease. MRD3 shows a throughput of 18.6 Mbits/s, which is a 3× improvement over R2.2 and within 95% of the measured throughput in the original test area. These results suggest that the widely-separated MRD radios can provide better overall reception coverage and maintain higher throughput than the co-located MRD radios in our experiments.

TCP Performance

Although our experiments show that MRD-RD achieves high UDP throughput gains, Section 3.6.2 discusses how MRD-RD can introduce somewhat greater variation in packet delivery delay. Such effects could adversely affect TCP's performance. We conducted experiments to evaluate how well TCP performs on MRD-RD.

We ran the HIVAR experiments, using TCP to transmit 72.4 MB of data from the mobile laptop and measured the average TCP throughput for two non-MRD schemes, R1.2 and R2.2, and two MRD-RD schemes, MRD-R1 and MRD-R2. In MRD-R1, we configure the R1.2 and R2.2 radios to be the active and passive MRD radios respectively. In MRD-R2, we use the same radios but swap their active and passive roles. Note that our experiments run MRD-RD only in the uplink direction. The TCP receiver runs on the same machine as the RDC and uses a single radio (*i.e.,* the active radio) to send TCP acknowledgment packets (TCP-ACKs) to the laptop.

In all of the TCP experiments, the laptop transmitter moves within the original test area as illustrated in Figure 3-8. We observe that the single-radio R1.2 and R2.2 experiments transmitted at an average throughput of 8.10 Mbits/s and 3.67 Mbits/s (Figure 3.6.2). The results correspond to about 78% of the throughput in the UDP experiments. The decreased throughput is not surprising given the overhead and congestion-controlled behavior of TCP.

MRD-R1 and MRD-R2 clocked an average throughput of 15.2 Mbits/s and 13.2 Mbits/s respectively, which translates to a 1.88× and 1.63× improvement over R1.2. Like the single-radio experiments, MRD-R1 achieves 78% of the throughput of the corresponding UDP experiment (MRD3). Because MRD-R1 achieves about the same proportion of UDP throughput as the single-radio TCP experiments, we believe that TCP remains largely unaffected by the delay variations and other link-level interactions in MRD-RD.

Compared to MRD-R1, MRD-R2 achieves a lower proportion (68%) of MRD3's throughput. We have not analyzed our traces to determine the exact cause for the reduced gain but we found that the channel conditions between the laptop and R2.2 is quite poor. As a result, the TCP-ACKs could have suffered increased delay and losses in the reverse direction, affecting the sending rate in the forward direction.

Delay Analysis

Thus far, our analysis has focused on throughput and frame loss rates. A number of compelling wireless applications such as telephony and video streaming require a relatively low packet delivery delay not exceeding $100 - 150$ ms [54]. We analyze MRD-RD's delay performance here.

As described in Section 3.6.1, we insert a timestamp in the payload of a packet's first transmission attempt to measure the one-way packet delivery delay. Because it is difficult to synchronize PC clocks to within a few tens of microseconds,[13] we do not measure the one-way packet delivery delay. Instead, we measure the delay jitter above the minimum one-way packet delivery time d_i for packet i, which does not require clock synchronization between the sender and the receiver. Let s_i and r_i be the start and receive timestamps associated with packet i for all $0 < i < 100,000$ packets transmitted in an experiment. Then $d_i = r_i - s_i - \min_i(r_i - s_i)$.

We also applied a piecewise linear regression algorithm [72] to remove clock skew between the sender and the receiver (we measured clock drifts on the order of 50 microseconds per second). Note that we can compute the one-way packet delivery delay by adding $\min_i(r_i - s_i)$, which includes the nominal transmission time and processing delay. In practice, this number is less than one millisecond. We will ignore this minor adjustment and use the terms "delay jitter" and "delay" interchangeably.

Figure 3-15 shows the one-way delay distribution for our HIVAR experiments. The MRD-RD median delay is below 1 ms and has 25% more packets delivered than R1 and R2. The low median delay is due to its ability to maintain a high bit-rate throughout the experiments. However, about 35% of the packets in MRD-RD were delivered with a

[13]We require the fine clock synchronization granularity because the nominal transmission time of 802.11a at high bit-rates is less than 0.5 milliseconds.

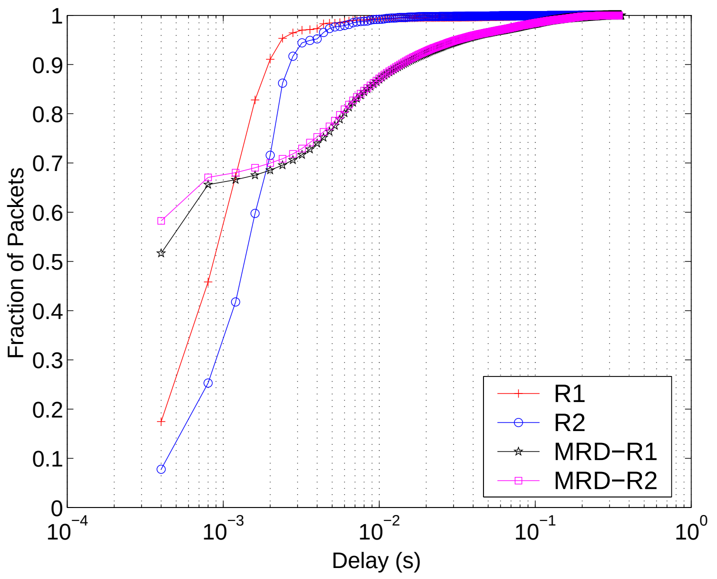


Figure 3-15: One way delay jitter for HIVAR experiments.

significantly higher delay than R1 and R2. Nonetheless, MRD-RD was able to deliver 95% of the packets within a delay of 35 ms, which is well below the delay bound of 150 ms that can be tolerated by telephony and video applications.

We attribute the increased packet delivery delay in MRD-RD to the fact that there were a significant number of frames that required retransmissions because our rate adaptation algorithm uses a set of aggressive minimum delivery thresholds to improve throughput (Section 3.4). In the design of the RDC, we assumed an in-order packet delivery service and added a ordering buffer at the RDC (Section 3.3.3). Whenever a retransmission is required, the ordering buffer blocks subsequent packets from being forwarded and increases the packet delivery delay for all of them.

Another source of delay comes from the losses of MRD-ACKs on the reverse channel, which delays the trigger to retransmit a packet. Also, the user-space implementation of the RDC is inefficient as interrupts and user-space buffering can add delays in generating and sending MRD-ACKs.

3.6.3 Low Channel Variability (LOVAR) Experiments I

We evaluate the performance of MRD-RD in a scenario where the channel variability is low, using the setup depicted in Figure 3-16. The parameters and methods we use for the LOVAR experiments are the same as the HIVAR experiments, except that we use a stationary desktop transmitter instead of a mobile one. The LOVAR experiments presented in this section are label as "LOVAR I". LOVAR I were conducted before we introduced our modifications to the MADWiFi autorate algorithm in Section 3.4. Thus, the autorate results presented in this section might understate the performance of the MRD-RD system. Nevertheless, the results of LOVAR I provides an interesting comparison of the system operating under different situations. In Section 3.6.4, we present the evaluation results of another set of LOVAR experiments (LOVAR II) that include the autorate enhancements for MRD.

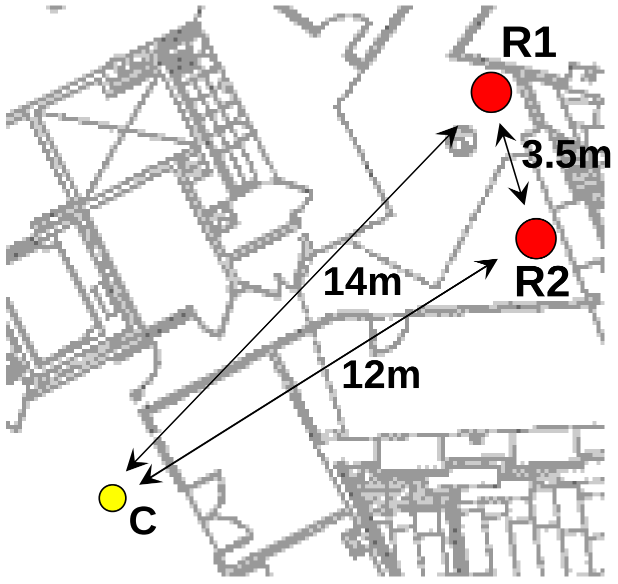
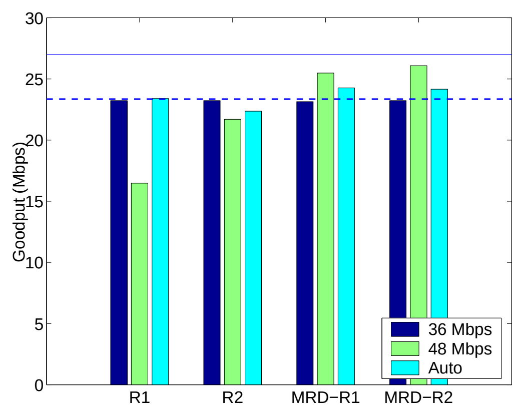
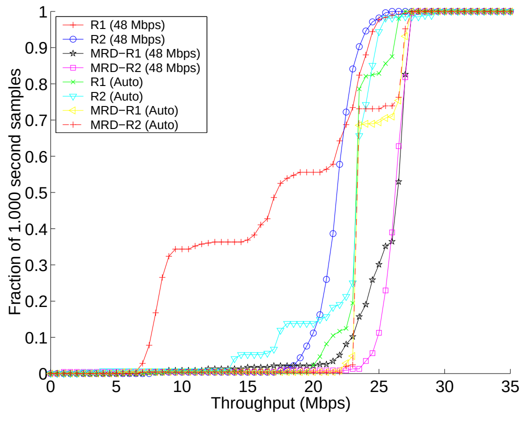


Figure 3-16: A diagram that illustrates the relative positions of the transmitter C and the receivers R1 and R2 in the LOVAR I experiments.



(a) Average Throughput



(b) One-second throughput distribution

Figure 3-17: LOVAR I Throughput Analysis. Left: Throughput averaged over 5 trials. The dashed line marks the maximum achievable UDP throughput (23 and 27 Mbits/s) for 802.11a fixed at 36 Mbits/s and 48 Mbits/s bit-rates. Right: Distribution of throughput averaged over non-overlapping one-second window samples.

Throughput

Figure 3-17(a) shows the throughput averaged over five trials for the LOVAR I experiments. We ran different experiments using two different fixed bit-rates (36 and 48 Mbits/s) and using the standard rate adaptation algorithm (Auto) implemented in the MADWiFi WLAN driver. The figure shows that the MRD-RD schemes at fixed bit-rate of 48 Mbits/s performed better than all other schemes. The dashed lines marks the maximum 802.11a UDP throughput for a fixed bit-rate 36 and 48 Mbits/s links, which are 23 and 27 Mbits/s

Experiment	FLR	FRR	FRR_{SS}	FRR_{FC}
MRD-R1	0.359	0.895	0.694	0.200
MRD-R2	0.354	0.958	0.819	0.139

Table 3.3: Frame loss (FLR) and frame recovery rates (FRR) of the low channel variability experiments. FRR is decomposed into two sources of recovery: soft selection (FRR_{SS}) and frame combining (FRR_{FC}).

respectively. The MRD-RD throughput is between 94.4% and 96.6% of the maximum UDP throughput at a bit-rate of 48 Mbits/s. Despite the overhead of transmitting MRD-ACK packets, MRD-R1 increases throughput over R1 by 54.6% at the fixed bit-rate of 48 Mbits/s, while MRD-R2 improves throughput over R2 by 20.2% at 48 Mbits/s. Similar to the HIVAR experiments, both MRD-R1 and MRD-R2 achieved very similar throughput results, again suggesting that the performance of MRD-RD is relatively insensitive to the choice of active AP in our experiments.

Under autorate (Auto), the throughput gains by MRD-R1 and MRD-R2 diminish to 3.7% and 8.1% respectively. One possible reason for the diminished gains is that the LOVAR I MRD-RD experiments used the unmodified version of the autorate algorithm. The algorithm ignores information from the MRD-ACK, so it adapts its bit-rate based only on the observed loss rate of the link layer transmissions to the active AP. Consequently, the algorithm selects a suboptimal bit-rate. For example, Figure 3-17(b) shows that MRD-R2 (Auto) selected 36 Mbits/s roughly 70% of the time even though our fixed rate experiments shows that it could achieve a high throughput at 48 Mbits/s.

Another reason for the diminished gain is the low variability of the channel. Although the frame loss rate was substantial at 48 Mbits/s, there was almost no loss at 36 Mbits/s. Thus, the throughput for $R1$ and $R2$ is lower bounded at the 36 Mbits/s bit-rate and caps the maximum achievable throughput gain for MRD-RD.

Source of Improvement

We analyze the sources of improvement for the 48 Mbits/s fixed bit-rate LOVAR I experiments and summarized the results in Table 3.3. The active APs in the LOVAR I experiments observed similar frame loss rates to the ones observed in the HIVAR experiments, but the FLR is much higher. It ranged between $90\% - 96\%$ for LOVAR I compared to $50\% - 57\%$ for HIVAR. There were also a larger number of frames recovered by frame combining in the LOVAR I experiments.

We found that the total number of frame combining attempts was proportionally similar to the HIVAR experiments. 37% and 25% of the total number of frames that were not successfully received by the active AP in MRD-R1 and MRD-R2. Thus, the increased number of frame combining recoveries was caused by a large reduction in the frame combining failure rate. Indeed, the frame combining failure rate p_f was about 45% in both sub-experiments, which is a large drop from the 80% in the HIVAR experiments.

Like the HIVAR experiments, the average p_f drops dramatically if the frames were subdivided into smaller blocks. Our simulation shows that $p_f = 17\%$ for $N_B = 91$ (i.e., $B = 16$ bytes). At the same time, Δ_{succ} (defined in Section 3.6.2) remains low for the successfully combined frames: the 95th percentile of Δ_{succ} for $N_B = 91$ is 10.

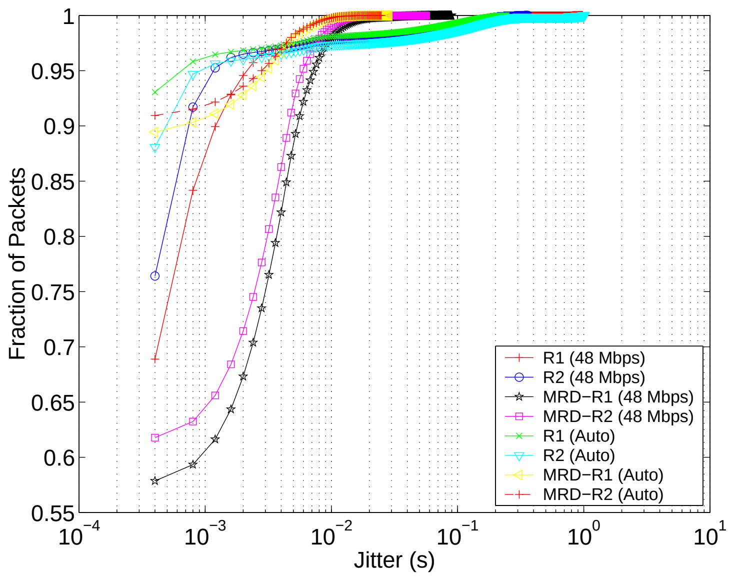


Figure 3-18: One way delay jitter for the LOVAR I experiments.

Delay Performance

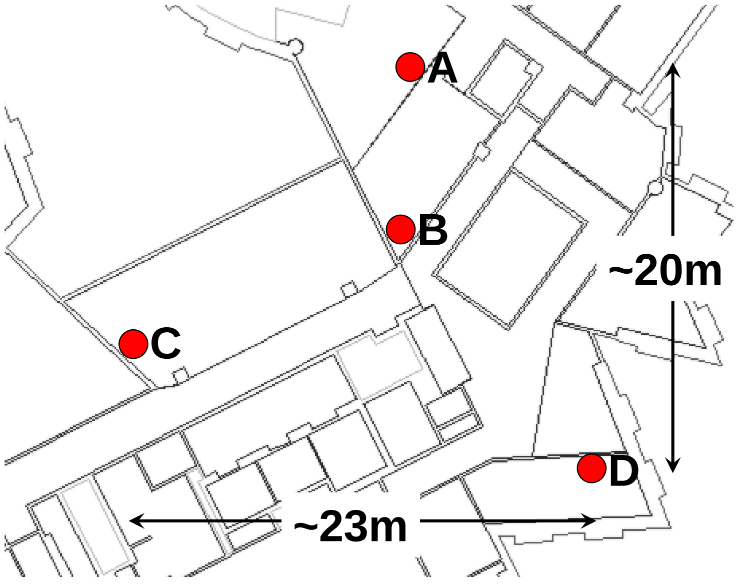
We repeat the delay analysis in Section 3.6.2 for the LOVAR I experiments. Figure 3-18 shows the one-way delay distribution for the fixed and autorate experiments. Compared to the HIVAR experiments, MRD-RD delivered packets with a lot smaller delay because it was able to recover almost all corrupt frame transmissions to the active AP. As a result, the LOVAR I experiments required much fewer frame retransmissions than the HIVAR experiments. Our LOVAR I experiments show that MRD-RD delivered 99% of the successfully received frames within 20 ms.

Finally, we observe a long tail in the one-way delay distribution (but representing only a tiny fraction of the transmitted packets) for the single-radio schemes that last up to several hundred milliseconds. This tail is mostly an artifact of handling retransmissions in the driver, where kernel interrupts can happen in between retransmissions. We suspect that the long tails arise from packets that require a large number of retransmissions because such packets may experience more interrupts from the kernel than packets delivered with fewer or no retransmissions. The MRD-RD scheme does not have this long tail because it successfully delivers packet with a lot fewer (re)transmissions.

In summary, we found that MRD-RD produced throughput gains in all experiments, regardless of the channel variability experienced by the client. In our HIVAR experiments, we found that MRD-RD was able to increase throughput up to three times that of the best AP when only a single radio is used. In our LOVAR I experiments, MRD-RD, at a *fixed* bit-rate of 48 Mbits/s, the throughput improvement is less impressive but still noticeable, between 9% and 16%.

3.6.4 Low Channel Variability (LOVAR) Experiments II

We ran another set of LOVAR experiments, called LOVAR II, that use stationary nodes and include the enhanced rate adaptation algorithm for MRD (Section 3.4). The parameters



(a) Setup.

Config.	R1	R2	C
I	C	D	A
II	C	D∗	A
III	C	D	A∗
IV	C	B	A
V	D	B	A
VI	C	D	B
VII	C	A	B
VIII	A	D	B
IX	A	B	C
X	A	B	D

(b) Node configurations.

Figure 3-19: Left: The positions of the stationary nodes used in the LOVAR II experiments. Right: The columns R1, R2, C identifies respectively the nodes that were configured as R1, R2, and the stationary client transmitter in the indicated configurations. Nodes A and D are equipped with two radios but only one is active in an experiment. The "∗" indicates that the experiment used the alternate radio instead of the main one. There are ten configurations in all.

and methods we use for the LOVAR II experiments are the same as LOVAR I, except that *all* of the LOVAR II experiments use the enhanced rate adaptation algorithm. The experiments were conducted throughout various times of the day.

LOVAR II includes ten different experiments with configurations that uses different combinations of stationary nodes shown in Figure 3-19(a). Table 3-19(b) lists the node combinations of each of the ten experiment configurations. While four nodes can generate additional client-and-receivers (R1 and R2) combinations, some configurations were left out because at least one of the client-receiver paths was completely out-of-range.

Table 3.4 summarizes the measured throughput averaged over five trials in each of the LOVAR II experiments. The results show that the average throughput gains of MRD over single-radio schemes are bimodal. The gains range from a factor of 0.99× to a factor of 9.2×, with 8 out of 11 sets of experiments showing MRD gains of less than 1.13×.

We have not been able to draw any substantive conclusions from these results, but we make two observations that motivate further investigations about our stationary node experiments. First, there seems to be little correlation between the balance of performance between R1 and R2 and the gain that MRD can achieve when it uses both paths simultaneously. As one would expect, paths with comparable performance should provide increased MRD gains (*e.g.,* Config. VI), while paths with extremely different performance should provide little or no MRD gain (*e.g.,* Configs. IX and X). However, there were some experiments that provided some MRD gain with extremely uneven paths (*e.g.,* Configs. III-1, IV, V), some experiments that used paths that were much less uneven (*e.g.,* Configs. VII, VIII) provided only marginal gains. Also, we repeated the Config. III experiments on a different

Config.	R1	R2	MRD-R1	MRD-R2	Gain
I	1.9 (1.2,3.3)	3.6 (2.9,4.7)	21.6 (21.0,22.1)	19.4 (18.4,20.3)	5.98
II	2.1 (1.6,2.6)	2.0 (1.2,2.8)	18.3 (16.2,20.5)	18.0 (15.9,20.7)	8.72
III-1	20.7 (17.1,21.9)	1.3 (0.7,2.2)	22.7 (21.1,24.2)	23.5 (23.0,24.2)	1.13
III-2	1.4 (1.1,2.0)	2.1 (1.9,2.2)	19.2 (16.7,22.2)	16.7 (15.9,18.6)	9.19
IV	1.5 (0.1,3.1)	25.1 (23.8,25.7)	26.4 (25.9,27.0)	24.3 (23.6,24.8)	1.05
V	2.9 (2.8,2.9)	23.7 (22.9,24.9)	25.8 (24.8,26.3)	24.2 (23.4,24.8)	1.09
VI	19.7 (17.9,21.3)	22.6 (21.2,23.8)	25.2 (24.4,25.6)	24.7 (24.1,25.0)	1.12
VII	18.4 (17.0,20.5)	26.1 (25.5,26.4)	27.4 (26.9,27.7)	26.1 (25.6,26.5)	1.05
VIII	25.8 (25.4,25.9)	20.2 (18.0,22.2)	26.4 (25.9,27.3)	26.9 (25.5,27.7)	1.04
IX	1.9 (1.2,2.9)	22.8 (21.0,24.7)	2.2 (1.9,2.6)	22.6 (20.3,24.3)	0.99
X	2.5 (2.4,2.6)	22.7 (22.3,22.9)	22.8 (22.6,23.1)	22.3 (22.1,22.6)	1.00

Table 3.4: Throughput for the ten stationary configurations in the LOVAR II experiments. Each entry shows the average (max, min) values of the five trials of each experiment set. The gain is defined as max(avg(MRD-R1), avg(MRD-R2))/max(avg(R1), avg(R2)). The experiments of Config. III were repeated on a different day.

day during our test runs and found that their results have changed significantly between the runs. Perhaps some objects in the environment have changed positions between the runs and caused the drastic change in our measurements.

Another aspect of our results that warrants further investigation is the high gain observed in Configurations I, II, and III-2, which are 6.0×, 8.7×, and 9.2× respectively. Table 3.4(a) shows the average link layer FLRs at different bit-rates for the single-radio and MRD schemes. The link layer FLRs[14] were measured at each of the individual receivers R1 and R2 (*i.e.,* nodes C and D) that were used in the Config. I experiments. Similar to our observations in Section 3.6.2, the low bit-rates did not effectively reduce the FLRs for the single-radio schemes. At a bit-rate of 24 Mbits/s, the FLRs for the single-radio reception schemes R1 and R2 are 39% and 77%. In contrast, R2 in the MRD-R1 scheme and R1 in the MRD-R2 scheme, *i.e.,* the passive MRD radios in the respective MRD schemes, have FLRs of 2.3% and 2.0% respectively. Because we measured the link layer FLR at the individual receivers, we do not expect their values to vary much between the single-radio and the MRD communication schemes. Yet, the individual receivers saw over an order of magnitude difference in the link layer FLRs between the two schemes!

To test whether the observed behavior is an artifact of a malfunctioning wireless interface, we conducted experiments Configs. II and III-2. These experiments use the same set of nodes as Config. I, except that they use an alternate radio interface[15] installed on the transmitting and receiving nodes. In particular, R2 receives transmissions from its alternate radio in Config. II. In Config III-2, the client (C) uses its alternate radio to transmit frames. Tables 3.4(b) and 3.4(c) shows that Configs. II and III-2 follows the same link layer FLR trend as Config. I. Thus, the abnormal FLR difference between the single-radio and MRD experiments still exists even when data is transmitted and received via a different wireless

[14]The link layer FLRs are different from the MRD FLRs: The former is measured at the receiver and the latter is measured at the MRDC after frame combining and soft selection.

[15]The antenna of the alternate radio is separated by about 50 cm from the main radio, as shown in Figure 3-13(a).

(a) Config. I

Bit-rate (Mbits/s)	R1	R2	R1 (MRD-R1)	R2 (MRD-R1)	R1 (MRD-R2)	R2 (MRD-R2)
6	0.694	0.401	$*$	$*$	$*$	$*$
9	0.375	0.264	$*$	$*$	$*$	$*$
12	0.475	0.383	$*$	$*$	$*$	$*$
18	0.440	0.386	$*$	$*$	$*$	$*$
24	0.768	0.387	0.861	0.023	0.020	0.578
36	0.837	0.418	0.619	0.016	0.091	0.483
48	1.000	0.727	0.964	0.562	0.944	0.745
54	1.000	0.980	1.000	0.991	1.000	0.982

(b) Config. II

Bit-rate (Mbits/s)	R1	R2	R1 (MRD-R1)	R2 (MRD-R1)	R1 (MRD-R2)	R2 (MRD-R2)
6	0.629	0.661	$*$	$*$	$*$	$*$
9	0.677	0.587	$*$	$*$	$*$	$*$
12	0.716	0.612	$*$	$*$	$*$	$*$
18	0.669	0.593	$*$	$*$	$*$	$*$
24	0.698	0.574	$*$	$*$	0.022	0.790
36	0.771	0.584	0.725	0.028	0.343	0.747
48	0.992	0.954	0.987	0.882	0.738	0.934
54	1.000	0.999	1.000	1.000	0.960	1.000

(c) Config. III

Bit-rate (Mbits/s)	R1	R2	R1 (MRD-R1)	R2 (MRD-R1)	R1 (MRD-R2)	R2 (MRD-R2)
6	0.749	0.620	$*$	$*$	$*$	$*$
9	0.763	0.660	$*$	$*$	$*$	$*$
12	0.786	0.622	$*$	$*$	$*$	$*$
18	0.765	0.637	$*$	$*$	0.031	0.744
24	0.772	0.615	$*$	$*$	0.029	0.663
36	0.804	0.627	0.819	0.025	0.161	0.708
48	1.000	0.949	0.993	0.801	0.983	0.974
54	1.000	1.000	1.000	0.998	1.000	0.998

Table 3.5: The FLRs (expressed as a ratio between 0 and 1) observed at R1 and R2 at different bit-rates. The right-most two pairs of columns are the raw FLRs observed at the individual receivers R1 and R2 during the MRD-R1 and MRD-R2 experiments using Config. I. In Configs. I, II, and III-2, the MRD schemes never had to reduce the transmission bit-rates below 24 Mbits/s, 24 Mbits/s, and 18 Mbits/s respectively.

interface.

We know that the MRD communication schemes configure the receivers to operate in different modes. The active radio operates as an access point in the AP infrastructure mode while the passive radio operates in the monitor mode. The abnormality seems to affect only the passive radio in the MRD schemes but we have not been able to identify its cause. Therefore, we cannot conclude that MRD can achieve throughput gains of factors that are as high as 9.2×. We also note that the problems described in this section do not occur in any of our other experiments.

3.7 Discussion

We discuss various implications that the MRD-RD system has on rate adaptation, capacity, and contention control in wireless LANs.

3.7.1 Rate Adaptation

The conventional wisdom of managing link quality in WLANs is to have the clients adapt to the channel conditions (i.e., adapt the bit-rate) *before* changing to an alternate link (e.g., AP) with a better channel quality. MRD-RD can be viewed as taking the opposite approach, where the clients use multiple links simultaneously before changing their bit-rate to adapt to the underlying “diversified” channel.

Our experimental results suggest that, with MRD-RD, even a simple rate adaptation algorithm, such as the one based-on MADWiFi, can perform well in different environments. In Section 3.6, we observed a large performance difference between the HIVAR and LOVAR experiments in the non-MRD schemes. We believe that the large performance difference is attributed not only to the increased frame loss rates observed at the individual APs, but also to the sub-optimal bit-rates that might have been chosen by the MADWiFi autorate algorithm in the high channel variability environment. (In Section 3.4, we tuned the algorithm to work well with multiple radios and frame combining, but did not alter the fundamental mechanisms used in the algorithm.)

In fact, we can use the results from the previous section to show that there is room for improvement in the rate adaptation algorithm. Table 3-10(a) and Figure 3-10(b) show that the frame loss rate to the active AP in MRD-R1 was 35% and that MRD-R1 selected a bit-rate of at least 24 Mbps over 90% of the time. Multiplying $1 - FLR$ with the the effective throughput of the 24 Mbps (17.8 Mbps) bit-rate yields 11.6 Mbps. Thus, we could have fixed the bit-rate to 24 Mbps for the non-MRD HIVAR experiment ($R1$) to improve the performance by 1.4× over the MADWiFi autorate algorithm, which achieved 8.25 Mbps. (Although the improvement is significant, it is not as great as MRD-R1, which achieved a 2.3× improvement at 18.7 Mbps.)

We are not suggesting that a fixed bit-rate should be used for non-MRD wireless links operating in a channel with high variability: selecting an optimal fixed bit-rate for such a channel still requires an adaptive algorithm. Rather, our intent is to use the example to motivate the following open questions: 1) Could other existing autorate schemes (e.g., RBAR [44], AARF [55], MiSer [76], OAR [80], SampleRate [24]) be used to improve performance of the non-MRD schemes in our HIVAR experiments? 2) Can we design an autorate algorithm for a non-MRD WLAN that performs well under a variety of channel conditions in a real environment that produces different types of errors (such as those observed in Section 3.6.2)? These are open questions, but we have demonstrated—using real-world experiments—that MRD-RD can use a simple rate adaptation algorithm to produce good performance under different and difficult channel conditions, and that with MRD-RD, the need for a finely tuned rate adaptation algorithm is not as important as with schemes that use single-radio terminals.

Finally, we should consider developing a rate adaptation scheme that can help reduce packet delivery delay for MRD. Because retransmissions in MRD requires higher delay than conventional scheme, as shown in Section 3.6.2, we should modify the rate adaptation algorithm (*e.g.,* adjust the minimum delivery threshold) to reduce retransmissions. Although doing so might trade off throughput gains for reduced delays, we believe that it is possible to develop an algorithm that provides both throughput gains and reduced delays in MRD because of MRD's fundamental ability to reduce frame loss rates and their variations.

3.7.2 Capacity

One may raise the question whether the overall capacity of a wireless network drops with MRD-RD since an MRD-RD system sacrifices the possibility of channel reuse at different APs for the sake of increased reliability for individual sessions. Let us consider a simple scenario with two terminals, T1 and T2 and two APs, R1 and R2. In the non-MRD system, each terminal is assigned to a single AP and transmit in orthogonal radio channels. In the MRD-RD system, we equip R1 and R2 with multiple radios such that each can receive transmissions from T1 and T2 at the same time. Thus, we have two MRD-RD sessions, T1-(R1,R2) and T2-(R1,R2) running simultaneously over different channels. In this case, it is clear that the MRD-RD system has a higher capacity than the non-MRD system since it has two extra transmissions (T1-R2 and T2-R1) over the non-MRD system without extra cost or interference to the existing transmissions, T1-R1 and T2-R2.

The results are similar if we assume that all nodes operate in the same radio frequency and that the two pairs T1-R1 and T2-R2 are close to each other (i.e., adjacent cells) such that simultaneous transmissions from both terminals cause significant interference to each other. Then under any channel access scheme that avoids simultaneous transmissions (mutual interference) from the two transmitters (*e.g.,* CSMA), the MRD-RD system will have a higher capacity than the non-MRD system. Like the previous scenario, the MRD-RD system here provides extra complementary transmissions (T1-R2 and T2-R1) to improve the reliability of the communication between the terminals and the APs.

However, the results are different if we allow T1-R1 and T2-R2 to transmit in different channels in the non-MRD system but constrain R1 and R2 to operate in the same channel in the MRD-RD system.[16] In this case, the total capacity for the non-MRD system is higher than the capacity of the MRD-RD system because simultaneous transmissions are possible in the non-MRD system whereas only one terminal may transmit at a time in the MRD-RD system due to the terminals' mutual interference.

Nonetheless, one can observe in Fig. 3-13(b) that the throughput of a single terminal in the MRD-RD system is higher than the combined (aggregate) throughput of T1-R1 and T2-R2 for the non-MRD system. Assuming that the combined throughput of the single-radio experiments closely approximates the aggregate throughput of the simultaneous communications (T1-R1 and T2-R2) in the non-MRD system, our experimental results suggest that the MRD-RD system can sometimes outperform the non-MRD system even though the capacity of the latter system is higher.

[16]This setup is a simplification of the following problem: When a network operator adds *single-radio* APs in a WLAN, is it better to run MRD-RD by configuring them as passive radios or is it better create a new cell by configuring them as a regular AP?

As discussed in the previous section, we believe that the reason for MRD-RD's higher *achieved throughput* is the better utilization of the resources with the MRD-RD system. In fact, MRD-RD not only reduces the frame loss rate, but also dampens the fluctuations in the frame loss rate. When the channel parameters are more stable, the rate adaptation algorithm does a better job in exploiting the existing bandwidth in the system. Hence, even though the non-MRD system has a higher capacity, it may not be utilized efficiently due to more variable conditions in each channel.

In the general case of more than two terminal-AP pairs, the capacity analysis is fairly complex and is beyond the scope of this dissertation. However, based on the examples given in the previous paragraphs, we can conclude that if the number of channels is sufficiently high, MRD-RD can increase the capacity of the network, since we can set extra "connections" between a terminal and an AP that are not directly paired, without causing any interference to the existing transmissions of the AP. Moreover, we illustrated an example in which MRD-RD leads to a higher achieved throughput despite a theoretically lower capacity, because MRD-RD provides less variable channel conditions that lead to better adaptiveness of the rate adaptation algorithm and better utilization of the wireless medium.

3.7.3 Link layer contention control

As mentioned in Section 3.5, the RDS needs to assume control over all retransmissions. Performing software-based retransmissions in the driver, however, also has the side effect of disabling the exponential backoff controlled by the firmware.

We acknowledge that the relative throughput improvement by MRD-RD may be reduced when exponential backoff is enabled. That is because the link layer increases the backoff window whenever a client fails to successfully transmit a data frame to the target receiver (i.e., the active AP) at the link layer. In our current design, the link layer is oblivious to MRD-RD. Even if the data frame is recovered through soft selection or block-based combining, the link layer may not reduce the contention window (which is what CSMA does when the link layer transmission succeeds). Consequently, the backoff window may increase unnecessarily and reduce MRD-RD's performance.

We can alleviate the problem by creating an interface that allows MRD-RD to inform the link layer backoff mechanism about the results of frame recovery at the RDC. Designing a medium access control algorithm that can adapt to MRD-RD's error recovery results is an interesting open problem. As an alternative, we can adopt some of the recently proposed channel access methods that do not rely on link layer acknowledgments but adjust the contention window size based on the measured idle channel time [43, 86].

Despite the above caveat, MRD-RD effectively reduces frame losses and the total number of transmissions required to deliver a packet, without increasing the nominal frame transmission time as in other existing approaches like using lowering data rates or employing forward error correction.

3.8 Chapter Summary

MRD-RD uses wireless path diversity to improve loss resilience in wireless local area networks. It coordinates wireless receptions among multiple radios—either co-located on the same device or distributed across different access points in the WLAN infrastructure—to increase loss resilience against path-dependent corruptions in the wireless medium. Using multiple radios, MRD-RD performs frame combining, which attempts to correct bit errors by combining corrupt copies of data frames received by each radio in our system. Because losses are often independent among different receivers, MRD-RD is able to achieve significant improvement in loss rates.

Our experiments in an in-building testbed using commodity PCs and 802.11a/b/g wireless interfaces demonstrate throughput gains between 1.9×-3.0× that of single-path communication schemes, under an environment with high channel variability in the uplink direction. The corresponding one-way delay bounded to 35 ms for 95% of the delivered packets.

From the experience we gathered in building and evaluating MRD-RD, we discovered a number of performance optimizations such as marking packets for low-latency and out-of-order delivery, and sharing MRD-RD feedback with the link layer to improve rate adaptation and contention window adjustments.

Chapter 4

MRD-Transmit Diversity (MRD-TD)

In the previous chapter, we described how MRD uses multiple receivers to improve the efficiency of delivering packets in wireless LANs without consuming much extra overhead. This chapter describes how MRD uses multiple transmitters (transmit diversity) to achieve the same goal.

MRD's transmit diversity (MRD-TD) sub-system is based on the following intuition: the process of choosing an appropriate transmission path (transmit radio) to deliver link layer data frames to and from a client needs to adapt to short-term channel variations to obtain good performance. Our measurements (detailed in Section 4.1) suggest that fine-grained path selection for each frame transmission to client stations can substantially reduce the number of transmissions required to successfully deliver a packet to a receiver.

There are two reasons why such fine-grained control is effective:

1. Frame losses occur in bursts, and many of these bursts are of long lengths on the order of tens of frames, implying that the conditional probability of losing a frame given that one had been lost in the recent past is often significantly larger than the average frame loss rate.

2. Three of the five main causes of frame losses mentioned in the beginning of Chapter 1—obstacle attenuation, multipath, and mobility—depend on the *path* traversed between an AP and a client. Thus, the choice of transmit radio and the client's location can significantly affect performance. Furthermore, channel contention near a transmit radio (*e.g.,* a given client's AP) may prevent it from sending a frame even when there is no contention near the receiver, yet another property that depends on the choice of transmit radios.

These observations motivate a WLAN data distribution system that permits fine-grained client-specific path selection among a set of neighboring transmit radios or APs. MRD-TD attempts to choose a transmit radio based on short-term frame delivery statistics, with the goal of adapting to short-term variations using path diversity.

The main challenge that our MRD-TD design overcomes is high path switching cost. Switching transmission paths in conventional WLANs requires a sequence of message exchanges to authenticate and register information about a client, which cannot not happen frequently because they cause large interruptions in transmission flow. To support fast and efficient path switching among transmission sites, MRD-TD uses a central controller to manage authentication and registration so that switching paths no longer requires negotiation between the transmit radios and the receiver. MRD-TD runs in conjunction with a longer-term primary-AP selection mechanism (for downlink transmissions), usually a card-specific proprietary mechanism, and can also be used with techniques for coping with high frame loss rates such as packet fragmentation [90], varying packet size [69], forward error correction (FEC) [60], adjusting data transmission rates [44], and mechanisms for improving performance in multi-rate WLANs [80, 87].

We present a fine-grained path selection heuristic that can reduce the average frame loss rates without consuming any extra bandwidth in the wireless medium. Using this heuristic, our prototype system reduces the average frame loss rates by as much as 26% compared to a fixed-path scheme that uses the best available path when receiver is in motion. MRD-TD also improves the transmission delay distribution by avoiding long burst losses. Because the two observed facts mentioned above are prominent for a moving client, we find that the benefits of MRD-TD are especially significant for such clients.

The rest of the chapter is organized as follows: Section 4.1 illustrates the benefits of transmit diversity by analyzing the loss correlation and the impact of localized interference in an experimental study that involves two transmitters and a single receiver. Section 4.2 describes the design and implementation of MRD-TD. Section 4.4 describes several performance results measured on our testbed based on 802.11b. We summarize the chapter in Section 4.5. Much of the work in this chapter also appears in [68] and builds on the results in [66], which shows, using trace-driven simulations, how transmit path diversity can improve the quality of low-latency video streams over 802.11 networks.

4.1 The Case for Fine-Grained Path Selection

We present experimental evidence and examples to make the case for fine-grained path selection. First, we gather measurements to show the short-term loss characteristics of an 802.11b testbed deployed in our building. Our results confirm that frame losses occur in bursts and reveal that the losses have little spatial correlation among different transmission sites (For uplink traffic, the transmission sites are the different transmit radios installed on a receiver. For downlink traffic, the sites are nearby APs.) Moreover, we find that when a frame loss occurs, the short-term probability of losing a subsequent frame transmitted from the same site is substantially greater than the short-term probability of losing a subsequent frame if it were sent from another site. We exploit this observation and design a system that seeks to avoid burst losses through fine-grained path selection.

We also analyze delay measurements from two concurrent packet streams transmitted from different sites, and show how localized interference causes intermittently high transmission delays. Our results demonstrate that transmission performance depends on path selection and further supports our case for fine-grained path selection.

4.1.1 Experimental Setup

Our setup, shown in Figure 4-1, consists of two 802.11b transmitters, A and B, and a receiving station placed at three different positions, R_1, R_2, and R_3 inside our lab. We conducted our experiments in late evening to avoid interacting with the building's daily 802.11 activity. All nodes are configured to run in the 802.11b *ad hoc* mode. A central packet generator sends a constant bit rate stream to the two wireless transmission sites via a 100

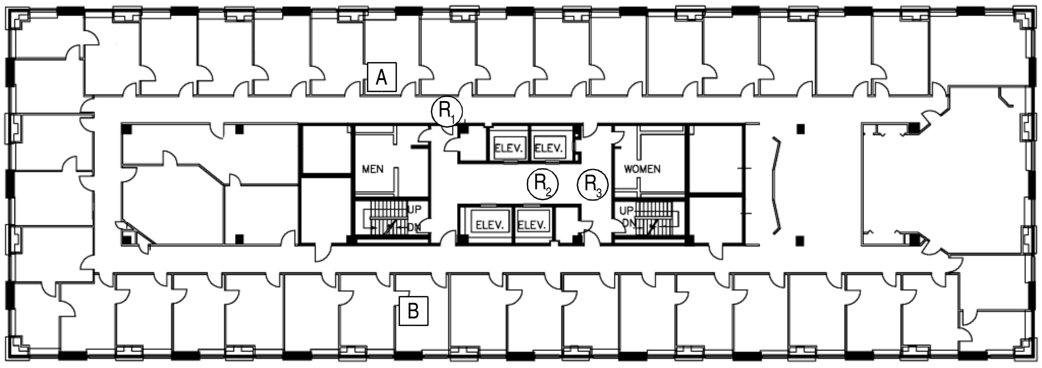


Figure 4-1: Floor-plan of the experiment setup. 802.11b transmitters at locations A and B each broadcast packets at 2.88 Mbits/s to a receiver at R_1, R_2 or R_3.R_3 is located in the middle of an elevator lobby where the walls are made of concrete and is approximately 15 meters from each transmitter.

Mbits/s wired link. For each packet it receives from the packet generator, each wireless site broadcasts the packet on its wireless interface. The packet generator is precisely calibrated to alternate packet transmissions successively between the two wireless transmitters to reduce potential collisions between them. The queues at each AP are large enough to prevent any losses due to buffer overflows.

The packet generator sends a stream UDP/IP packets at 240 packets per second to each wireless transmitter. Thus, the aggregate throughput of two transmitters is 5.96 Mbits/s, which is similar to the observed saturation throughput in [18]. We use a high packet rate to sample changes in the channel accurately. We use broadcast packets to measure the link layer data frame loss rate; broadcast packets avoid the effects of link layer retransmissions and exponential back-off delays, and the effects of link layer acknowledgment frame losses in the reverse direction. Because frame loss rates tend to decrease with frame sizes [69], we use 1500 byte packets, a maximum transmission unit commonly used in Ethernets, in all of our experiments.

We conducted two sets of experiments, *static* and *mobile*, done in separate trials, to examine the effects of stationary and mobile receivers. We conducted experiments at each of the three different receiver positions during quiet hours to ensure the channel is not affected by the building's daily activities. For the mobile experiments, the receiving laptop was carried by a human subject moving with random motion over a small area (2 m $\times$ 2 m) centered at each receiver location at a normal walking speed. There is no line-of-sight between the transmitters and the receiver. Such a location can be harsh for signal propagation, but it is not unrealistic. Each experiment transmitted $144,000$ frames in 5 minutes. The results presented are the averages of three trials. We show the results for $R3$ but the trends for all three receiver locations are very similar.

4.1.2 Burstiness and Spatial Correlation of Losses

We measure the loss characteristics of two concurrent packet streams transmitted from two 802.11b devices at different locations. We are interested in i) how bursty losses are, ii) how frame losses from different transmitters are related, and iii) how receiver motion affects loss characteristics.

Experiment	static		mobile	
Sender	A	B	A	B
$FLR(\%)$	4.79	10.2	17.1	15.3
$BLR(\%)$	1.5	4.6	10.8	9.2
BLR/FLR	31%	45%	63%	60%

Table 4.1: The average frame loss rate (FLR) and average burst loss rate (BLR) for the static and mobile experiments.

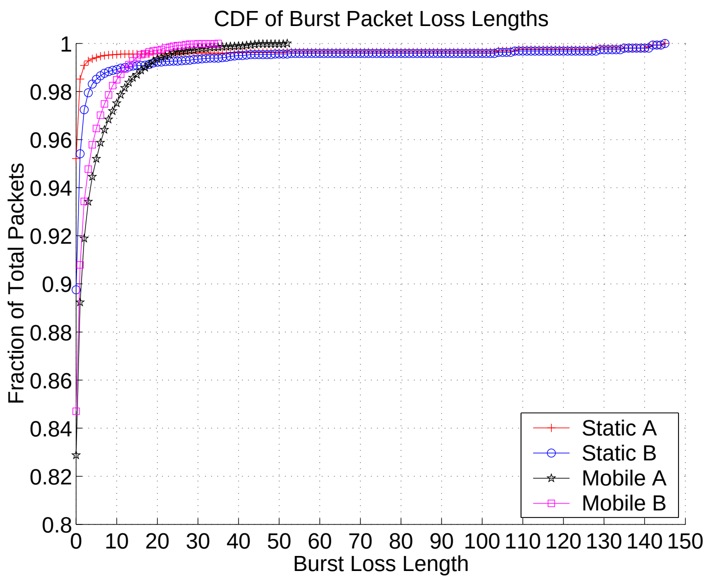
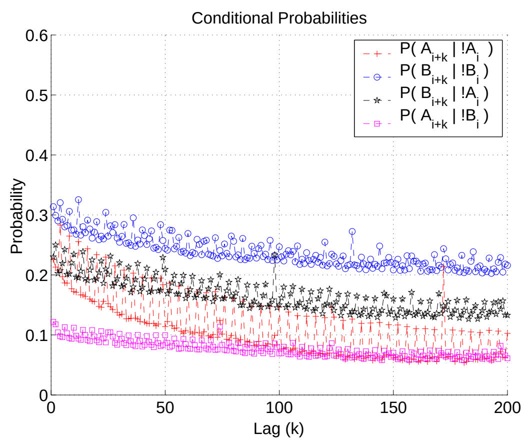


Figure 4-2: CDF of the length of burst losses for both static and mobile experiments.

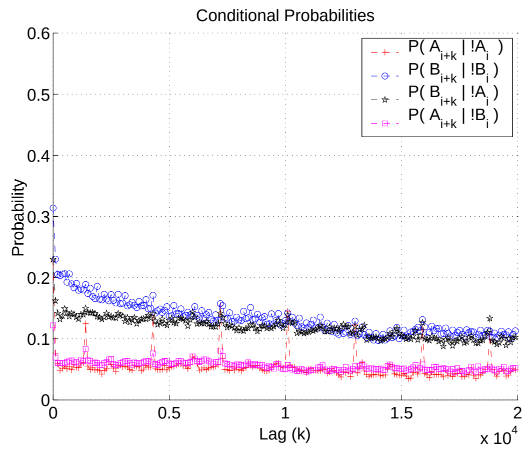
Table 4.1 shows the frame loss rate (FLR) and the *burst loss rate* (BLR) for each packet stream averaged over three trials in the static and mobile experiments at $R3$. The BLR is the number of frames lost in a burst of two or more consecutive frames divided by the total number of frames sent in the stream.

For each stream, our mobile experiment has higher FLR and BLR than the static experiments. The BLR/FLR ratio is greater than 50% for mobile and less than 50% for static, which suggests that our mobile experiment has more lost frames that occur in bursts than our static experiment. Figure 4-2 shows the CDF of burst loss length for each transmitter in both experiments. Although "static" has fewer lost frames that occur in bursts, the CDF shows that "static" has a long tail. When the receiver is static, losses can occur in a few very long bursts (up to 146). In contrast, the maximum burst loss length for "mobile" did not exceed 53. While we cannot pinpoint the exact cause for this behavior, we believe that a receiver's movement can lower the maximum burst loss length; a mobile receiver can move out of a bad location where the channel quality is extremely poor while a static receiver can suffer long bursts of losses due to sustained, problems in the transmission path between the static sender and receiver.

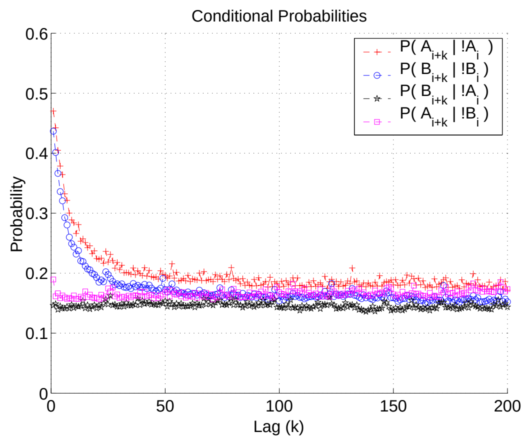
Next, we examine how frame losses are correlated between different transmitters (spatial) and at different times (temporal). Let A_i and B_i represent the lost of frame i sent from



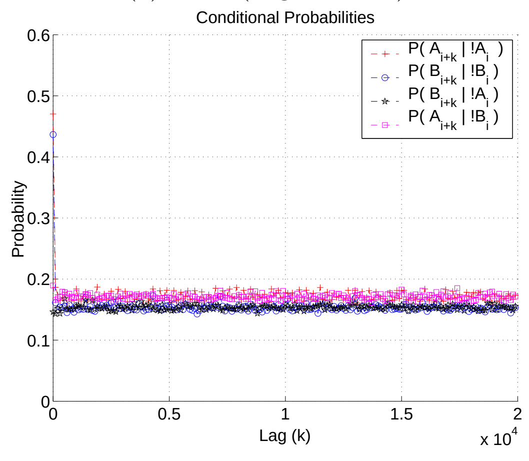
(a) Static (small k values)



(b) Static (large k values)



(c) Mobile (small k values)



(d) Mobile (large k values)

Figure 4-3: The auto-conditional and cross-conditional loss probabilities of frame losses at different frame lags k for the static (a) and (b) and mobile (c) and (d) experiments.

transmitters A and B respectively. Then, $P(A_{i+k}|A_i)$ and $P(B_{i+k}|B_i)$, for $k > 0$, represents the "auto-conditional loss probability" that the $(i+k)^{th}$ frame is lost, given that the i^{th} frame is lost in the same packet stream. If losses occur in bursts, we expect $P(A_{i+k}|A_i) > P(A)$, where $P(A) = FLR_A$. In contrast, if losses are memoryless or independent, we expect $P(A_{i+k}|A_i) = P(A)$.

Similarly, we use $P(B_{i+k}|A_i)$ and $P(A_{i+k}|B_i)$ to represent the "cross-conditional loss probability". Thus, if losses are correlated between the streams, we expect $P(B_{i+k}|A_i) > P(B)$, where $P(B) = FLR_B$. If losses are independent between streams, we expect $P(B_{i+k}|A_i) = P(B)$.

Figures 4-3(a) and 4-3(c) shows the auto-conditional and cross-conditional loss probabilities for the static and mobile experiments at small values of k, $1 \leq k \leq 200$ (4.2 to 840 ms).

In our mobile experiment, losses are bursty. Figure 4-3(c) shows that the auto-conditional loss probabilities ($P(A_{i+k}|A_i)$ and $P(B_{i+k}|B_i)$) are much larger than the respective average FLR (Table 4.1) for A and B at small lags. Thus, given that a frame loss occurs, the probability of losing the next few frames is much higher than the average FLR, which suggests that burst losses are likely to happen. In contrast, the cross-conditional loss probabilities ($P(A_{i+k}|B_i)$ and $P(B_{i+k}|A_i)$) remain nearly the same as the respective average FLR, which suggests that frame losses exhibit little correlation between the different transmission sites. Observe that the average FLR of A is larger than that of B, yet $P(A_{i+k}|B_i) < P(B_{i+k}|B_i)$. This suggests that fine-grained path selection can be effective in avoiding imminent burst losses by switching to an alternate site (path) whenever a loss occurs in the current site (path), even in cases where sites have different average FLR. *Thus, path diversity can effectively reduce time-correlated losses over time-varying wireless channels.*

In our static experiment, losses are less bursty than in our mobile experiment. Figure 4-3(a) shows that in almost all lags, $P(B_{i+k}|A_i) > P(A_{i+k}|A_i)$ but $P(A_{i+k}|B_i) < P(B_{i+k}|B_i)$. This is because the FLR of B is about twice that of A. While B can benefit by switching to A whenever a loss occurs, the converse is not true for A. Although fine-grained path selection is beneficial for our mobile environment, coarse-grained path selection based on long-term frame loss rates may be sufficient in our static environment (*i.e.,* when the channel is less dynamic). However, we will illustrate in the next section that in some cases, fine-grained path selection is beneficial for both static and mobile receivers.

Another interesting observation is that in our static experiment, $P(A_{i+k}|A_i)$ tends to be consistently higher for every k value that is a multiple of 4 (about 17 ms). Although we are not certain, we think that this behavior might be caused by collisions with beacon frames from nearby APs (belonging to the production 802.11b wireless LAN in our lab), each of which periodically broadcast beacons at every 100 ms. The superposition of several periodic beacon broadcasts with overlapping transmission range might produce the smaller (17 ms) periodicity observed in our graph.

Figures 4-3(b) and 4-3(d) show similar probabilities for our static and mobile experiments at large k values, $1 \leq k \leq 20,000$ (4.2 ms to 84 s). For clarity, we only plot data for each k value that is a multiple of 100. First, as k grows larger, the auto-conditional loss probabilities in both experiments converge to the corresponding frame loss rates. This behavior arises because frame losses become increasingly independent as the lag increases. We note that $P(A_{i+k}|A_i)$ is consistently higher for each k value that is a multiple of 3000 (about 12.5 s), but we are not certain of the cause.

4.1.3 Impact of Localized Interference

Most WLAN MAC protocols use a carrier sensing (CS) mechanism to reduce the likelihood of collisions. Before sending a frame, the sender senses the channel for activity. If the sender senses energy in the channel, it suppresses its transmission to avoid colliding with another potential ongoing transmission.

Carrier sense suppression depends on the relative positions of transmitting and receiving nodes, and can lead to the classical exposed terminal problem [23]. For example, when one AP's transmission is suppressed by CS, an alternate AP may be used to transmit data frames. However, the alternate AP's transmission cannot succeed if the interfering energy in the medium is too high at the receiver; the receiver must be at a location where the signal to interference ratio is sufficiently high (see Figure 4-4). A fine-grained path selection system can discover such transmission opportunities when they exist.

We gathered measurements that show that the scenario described above exists in a real network. During a busy hour, two transmitters, AP_1 and AP_2, alternatively send broadcast frames to a common receiver C. Table 4.2 shows that transmitter AP_1 offers

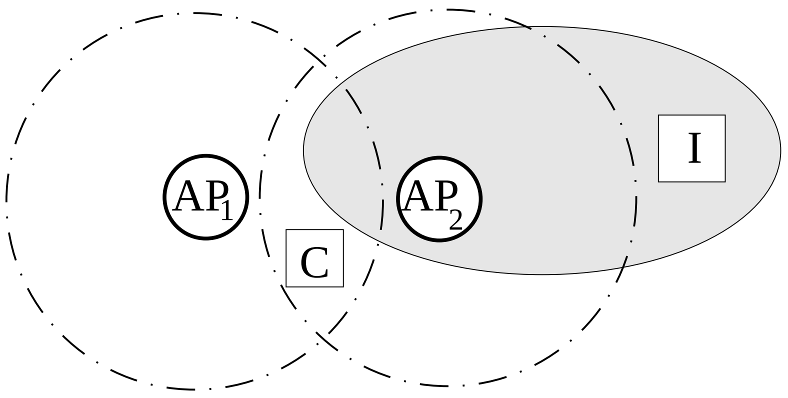


Figure 4-4: Carrier sense suppresses AP_2 from transmitting due to the interfering signal from I (*e.g.,* a WLAN client in another nearby network or an appliance that use the same frequency spectrum). However, AP_1 may be used to communicate with C because the interfering signal is not strong enough to affect either AP_1 or C.

Path	Loss Rate (%)	RSSI (dBm)	Jitter (ms)	DeferEngy (Count)
AP_1	1.82	-37.91	2.29	47389
AP_2	2.03	-44.46	0.35	22749

Table 4.2: Based on loss rate and average received signal strength, AP_1 is the preferred path. However, AP_1 has a much higher one-way delay jitter than AP_2.

both higher signal strength and slightly lower overall loss rate in a 30-minute packet trace of an experiment that involved transmitting 720,000 data frames. Thus, if AP_1 were an access point, the receiver would naturally associate with AP_1.

Figure 4-5 shows the received signal strength, the average loss rates of 1-second slices in the trace, and the one-way delay jitter (i.e., the delay variations above the minimum one-way delay value) as a function of time for a 60-second snippet of the packet trace. This 60-second snapshot is chosen to avoid cluttering the figure, and is representative of the characteristics manifested in the entire trace. The figure shows that the packet delay jitter from AP_1 is substantially higher than the delay jitter from AP_2. Lost packets were ignored from the delay jitter analysis. Because broadcast packets are not retransmitted and are not subject to the exponential back-off mechanism in 802.11 networks, the increased one-way packet transmission delays from AP_1 can only be attributed to the increased delay caused by the carrier sense mechanism (perhaps due to ongoing traffic from a nearby 802.11b network). We confirm this hypothesis by verifying the `DeferEngy` register in the transmitter's 802.11b interface, which counts the number of times that a packet has been deferred because energy was sensed in the carrier. Table 4.2 shows that the value of the `DeferEngy` register for transmitter AP_1 is much greater than the value for transmitter AP_2, indicating that transmitter AP_1 deferred transmission more than twice as many times as transmitter AP_2.

Table 4.2 shows that the loss rate for transmitter AP_1 is comparable to that of transmitter AP_2 even though the `DeferEngy` count for AP_1 is twice as large as AP_2. The result

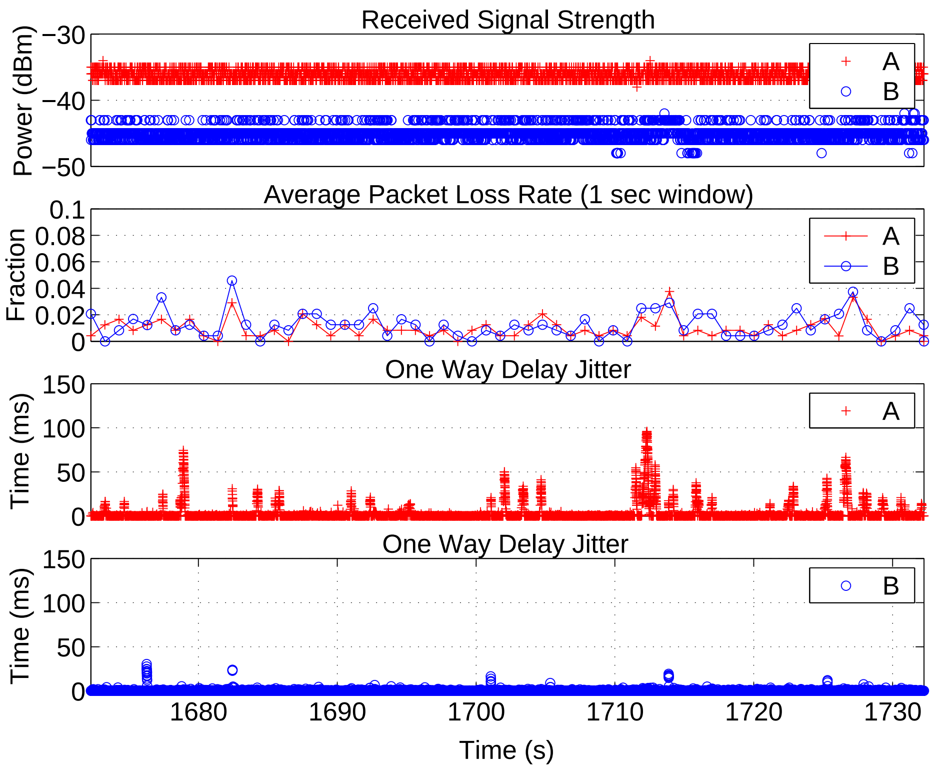


Figure 4-5: A 60-second snippet of a streaming experiment on two paths originating from transmitters AP_1 and AP_2. AP_1 (in red) offers higher signal strength and lower loss rate than AP_2. However, due to localized interference that triggers the carrier sensing mechanism in AP_1, AP_1 frequently suffers from high spikes of delay jitter, while the one-way delay from AP_2 remains low and relatively constant.

suggests that many of AP_2's transmission can succeed even when interfering energy is detected by AP_1. Thus, we have identified a real case of the example shown in Figure 4-4. *For these scenarios, fine-grained path selection can be used to reduce both loss and delay by switching data frame transmissions intelligently between the available APs (for downlink traffic) and between different transmit radios on a receiver (for uplink traffic).*

4.2 Design and Implementation of MRD-TD

To support downlink transmit diversity, MRD-TD deploys multiple APs within an area (Figure 4-6), interconnected over a wired network whose data rate is much higher than the wireless link rate, with each AP being able to detect whether a WLAN client is currently within its transmitting range or not (*e.g.,* using periodic probes). Optionally, MRD-TD can support uplink transmit diversity by installing multiple transmit radios on each client device. Both WLAN clients and APs use synchronous ACKs to immediately acknowledge every non-broadcast data frame received over the wireless link. This feedback is important, because it allows the data transmitter to determine path conditions at the granularity of individual frame transmissions.

MRD-TD uses a path-selection heuristic to determine which AP and which client trans-

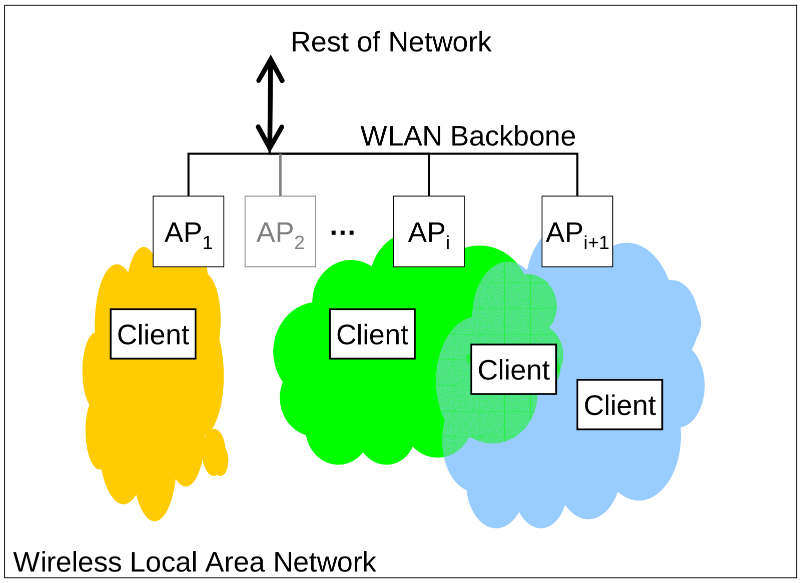


Figure 4-6: A cellular WLAN model where neighboring APs (*e.g.*, AP_i and AP_{i+1}) have overlapping coverage. To achieve the benefits of fine-grained path selection, MRD-TD requires that the client switch between APs quickly and at low cost. Section 4.2.2 shows how 802.11-like systems can achieve this goal.

mit radio, and hence which wireless path, to use for transmitting frames in the downlink and uplink directions. Section 4.2.1 discusses the details of the heuristic. MRD-TD also requires the ability to change the wireless path on a fine-grained basis without disrupting communication or incurring overhead. Section 4.2.2 describes how fast path switching can be achieved in cellular WLAN networks.

In a traditional WLAN architecture, the different APs deployed in a single WLAN require little explicit coordination between one another. On the other hand, MRD-TD requires explicit coordination because it makes path choices on a frame-by-frame basis (including sending frame retransmissions along a path different from the original transmission), with control over the path resting on the transmitter (AP) rather than on the receiver (client). To enable this coordination, MRD-TD extends the WLAN architecture by adding two components, the *MRD-TD Controller (TDS)* and the *MRD-TD Channel Monitor (CM)*, as shown in Figure 4-7. The TDS and CM run on the transmitter-side of the system—the wired backbone network and AP in the downlink direction, and the WLAN client in the uplink direction. In addition to the TDS and CM, we run a *MRD-TD Receiver (TDR)* process on the receiver to ensure packets are delivered in-order to the upper network layers.

The system works if TDS and CM components are deployed on only one of the two directions. In this case, fine-grained path selection is enabled in one corresponding direction only. Our current Linux implementation and experiments are for the downlink direction alone.

The TDS is responsible for forwarding each packet via one of the APs (or client transmit radios) that is within transmission range of the client (or of the associated AP for uplink traffic). The TDS runs a fine-grained path-selection heuristic, which makes a forwarding decision for each packet based on feedback sent by the CMs, each of which runs at an AP

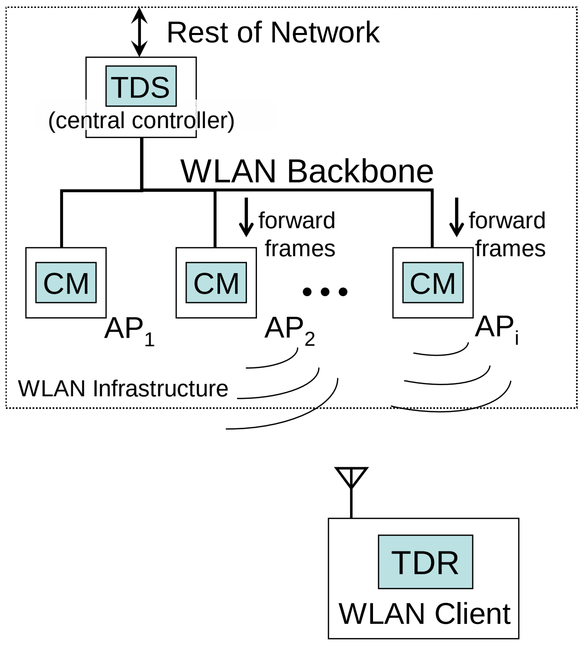


Figure 4-7: The MRD-TD architecture to perform fine-grained path selection among access points, shown in the downlink direction. The MRD-TD Controller (TDS) determines which path (AP) to use. Each AP runs a MRD-TD Channel Monitor (CM) that monitors link conditions and reports these conditions to the TDS. The client runs the MRD-TD Receiver (TDR) to ensure packets are delivered in-order to the upper network layers.

or transmit radio. A CM monitors the wireless link at the transmitter and sends two types of messages to the TDS, *registration event* messages and *path-condition update* messages.

Registration event The CM sends a periodic registration event to the downlink TDS whenever the CM detects that a particular client is within its downlink transmission range. The registration event allows the TDS to maintain a set of usable transmitters for fine-grained path selection. The event is maintained as soft-state at the TDS so that the registration can timeout when a client moves out of an AP's transmission range. The uplink TDS and CM use an analogous procedure to maintain a set of usable transmit radios for uplink transmissions.[1]

Path-condition update Each CM monitors the channel conditions in the direction of the data flow. In the downlink direction, the CM at the AP maintains this information per client. In uplink, the CM maintains information only for the associated AP. The CM periodically sends updates of this information to the TDS. The CM observes a failed transmission if the sender does not receive a synchronous ACK after a frame transmission. This failure can occur when either the data frame or the returning ACK is lost. The CM may also observe the receiver's received signal strength of the transmitted data frame if it is reported in the synchronous ACK. To reduce the

[1]Transmit radios installed on the same device are likely to have similar transmission range. In most cases, the uplink TDS and CM can simply assume that it can use all of the available transmit radios for fine-grained path selection.

overhead of reporting feedback to the TDS, the CM does not send per-frame level information to the TDS. Instead, it sends an update at regular intervals or whenever a threshold condition (see Section 4.2.1) has been satisfied.

To support retransmissions, the TDS wraps each data packet with a header that contains a field indicating the retransmission limit. If the sender (AP or client transmit radio) fails to successfully transmit a data frame to the receiver and receive an ACK, the corresponding CM decrements the retransmission limit field and returns it to the TDS for retransmission if the retransmission limit has not been exceeded. Because the TDS runs the path-selection heuristic for the packet, including those being retransmitted, the retransmission may be done along a different path.

The proposed retransmission scheme can introduce out-of-order frame delivery. To improve efficiency, the TDS does not wait for the status (the success or failure) of every frame transmission before transmitting the next frame, *i.e.,* transmissions are pipelined. Because the TDS decides when to retransmit a frame, the actual retransmission might occur after a successful transmission of another frame belonging to a subsequent packet in the transmission sequence. Some higher-layer network protocols such as TCP are sensitive to the ordering of delivered packets. To ensure in-order packet delivery, the TDR checks the sequence numbers of the delivered frames and buffers frames that arrived out-of-order until the missing packet arrives or a timeout occurs. We have not implemented pipelining nor the TDR for the stand-alone MRD-TD sub-system.

We made a deliberate design decision to put the path-selection decision control at the transmitter-side of the system. Alternatively, a receiver can monitor channel conditions and select different wireless paths. But receiver-side control has several drawbacks. First, the receiver can only detect lost frames from gaps in the sequence numbers of the transmitted frames, which means that it cannot detect a loss before it receives a successful transmission. When losses occur in bursts, a receiver may not be able to switch paths in time to avoid them. Second, when a receiver decides to switch paths, it must send a control message to notify the TDS to switch paths. Such a message is unreliable and is prone to loss when channel conditions are poor. None of these problems will occur when the transmitter makes path-selection decisions.

Moreover, a downlink TDS that resides in the distribution system has a global view of the wireless activities at all the different APs. Thus, the downlink TDS can, for example, measure traffic load among APs, track a client's movement, or detect which APs are suffering from localized interference from their carrier-sense mechanism, and adapt path-selection decisions accordingly.

4.2.1 MRD-TD Path Selection Heuristic

The goal of MRD-TD's fine-grained path selection heuristic is to reduce losses in the wireless medium without consuming extra wireless bandwidth. The heuristic, at any given time, selects only one AP/client transmit radio with a good transmission path to transmit a downlink/uplink data frame to/from a client. Our goal is different from techniques proposed in [84, 79, 75], which seek to aggregate bandwidth by using multiple orthogonal paths in parallel. Our goal is also different from schemes that employ forward error correction (FEC) across multiple paths. For example, a simple FEC scheme might replicate and transmit every data frame via all the APs that are within range of the client. Such schemes use redundancy to reduce loss rates in the wireless medium, while MRD-TD achieves the same goal through intelligent, fine-grained path selection without consuming extra wireless bandwidth.

In theory, a path-selection algorithm should select the best path for each data frame transmission. To do so, a system must acquire accurate knowledge of the wireless channel condition of each available path within a few milliseconds. In practice, accurate sampling of the channel conditions is difficult and might incur large overhead.

We observe that selecting the best path for every data frame is unnecessary to achieve good results. As observed in Section 4.1.2, frame losses usually occur in bursts, especially when the receiver is mobile, and different transmission paths often exhibit weakly correlated channel conditions. Therefore, a "reactive" path selection heuristic can be effective if it can determine whether the currently used transmission path has fallen into a bad state (*i.e.,* predict whether the next few frame transmissions will fail with high probability), and divert the subsequent transmissions to an alternate path. As long as the alternate path's average loss rate is not substantially higher than that of the current path, diverting the frame transmissions will likely avoid burst losses.path.

In MRD-TD, the CM running at each AP (or client) keeps track of the per-path history of the losses of the last data frames sent to each station within a window of H recent frame transmissions.[2] The CM then monitors the loss rate within this window. If the observed number of lost data frames is greater than a certain threshold T, the CM notifies the TDS to forward subsequent frames via a different transmitter. After a path switch occurs, the CM at the newly-selected transmitter waits for at least H data frame transmission attempts before signaling another switch to the TDS. Thus, H defines the switching time granularity (hysteresis), while T governs the sensitivity to the losses on the current path.

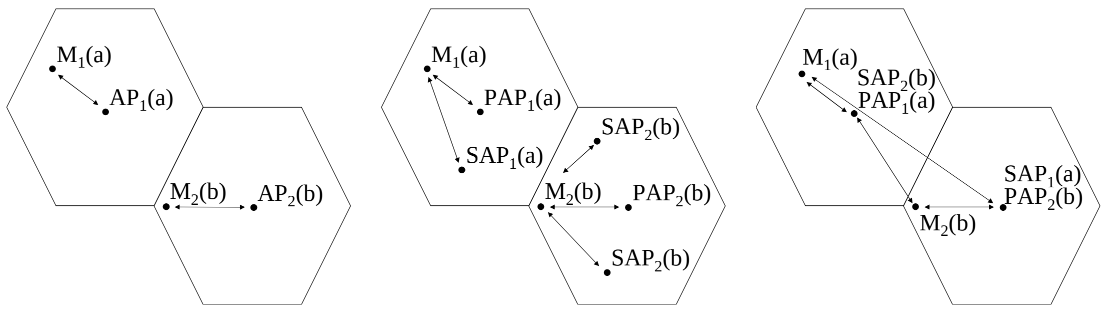
This heuristic is simple, and it uses feedback information only from the currently used transmitter. Active channel probing is unnecessary because the sender can detect a lost unicast frame by the absence of its synchronous ACK.

However, this heuristic is sub-optimal and will not work well under all channel conditions if the values H and T are fixed. A small value of H is desirable for bursty and dynamic channel conditions, so that the heuristic can adapt quickly. On the other hand, a larger value for H allows the heuristic to obtain a better estimate of the channel's average loss rate; a larger value is suitable under static channel conditions where the signal quality does not vary quickly. As shown in Section 4.1.2, often, a better selection strategy for static channel conditions is to "lock on" to the transmitter that has a lower average loss rate. Similarly, when the loss rates of the alternate paths are significantly higher than or are highly correlated with the current path, a large T is desirable to prevent switching to a potentially poorer path when only a small number of losses are detected in the current path. In other cases, a small T diverts packets early, which helps to avoid imminent burst losses in the current path.

Our experiments in Section 4.4 indicates that a choice of $H = 1$ and $T = 1$ works reasonably well for dynamic channel conditions when the receiver is mobile, and a choice of $H = 10$ and $T = 5$ works well when the channel is less dynamic. Fortunately, our experiments suggest that the observed loss rates are not too sensitive to the exact values of H and T. Nonetheless, we believe that the current heuristic can be improved by making H and T adaptive, *e.g.,* using simple machine learning techniques for learning parameters [71].

Finally, when the losses of a transmission path triggers a path switch, the heuristic has to

[2]The CM also needs to keep a timer (not implemented in our testbed) to flush the loss window so that H does not span a long time period when the traffic rate is low.



(a) Cellular WLAN (b) MRD-TD WLAN (c) MRD-TD WLAN (Low-cost)

Figure 4-8: Clients M_1 and M_2 belong to cells 1 and 2, which operate in channels a and b respectively. For (b) and (c), all APs in the same cell operate in the same channel. Arrows mark possible WLAN communication path(s) for each client. In a traditional cellular WLAN (a), clients only communicate with one AP. In the extended MRD-TD WLAN (b), the primary access points (PAP) associate with clients while the secondary access points (SAP) provide alternate communication paths within the same cell. A low-cost deployment alternative for MRD-TD is shown in (c), where the SAP is co-located with the PAP of another cell to reduce the number of AP deployment locations. In (c), M_1 has a long communication path to SAP$_1$ but in many cases, M_1 will communicate with a closer SAP located in another cell (not shown).

pick which one of the available alternate paths it should switch into. A simple mechanism is to randomly pick an alternate path from among a set of APs within communication range of the client. We currently assume that there is only one alternate path so our implementation always switches into that path whenever a path switch occurs.

We emphasize that the fine-grained path selection heuristic presented here is different from the handoff algorithms that are used to initiate a handoff process in many common cellular WLANs (described in the next section). Due to the high overhead in a typical handoff procedure, handoff algorithms often use a strong hysteresis to prevent a receiver from flapping handoffs among APs [28] when it finds multiple APs within range. In contrast, the MRD-TD heuristic can switch paths among APs on a frame-by-frame basis; thus, transmission paths are selected only as a function of channel conditions estimated by the per-client data frame loss history at each AP.

4.2.2 Reducing Path Switching Cost

For downlink transmissions, the MRD-TD design assumes that the WLAN incurs negligible cost when the transmission path is switched between different APs. This assumption is reasonable for WLAN architectures that support *soft handoff*. In a soft handoff, during the transfer of communication from one AP to another, a client maintains an undisrupted communication flow with both APs until the transfer completes. For example, code-division multiple access (CDMA) wireless networks support soft handoff, during which neighboring APs transmit signals simultaneously. Clients use RAKE receivers to resolve and decode the combined signals and maintain connectivity.

A WLAN that uses the same frequency channel for all its APs may also support soft-handoffs. However, typical WLANs such as 802.11 uses a *cellular* architecture (Figure 4-

8(a)) in which operators configure neighboring APs to use *orthogonal* channels to achieve spatial frequency reuse that increases the capacity of the network. In order to communicate with an AP in the network, a client needs to switch its communication channel to the one being used by the AP. Thus, in cellular WLANs, MRD-TD needs to explicitly notify the client to switch channels whenever it selects a new path for downlink communication. The overhead associated with switching paths can be significant (lasting from a few to hundreds of milliseconds [63]) especially when it occurs on a frame-by-frame basis. Moreover, forcing a client to communicate with an AP when the client is outside of that AP's cell boundary may increase co-channel interference and reduce the capacity of the network, as we will explain later.

One method of reducing the path-switching overhead is to install multiple radios on each client and statically associate each radio with a different access point that is within range of the client. This solution has two drawbacks: 1) it is not scalable; to take full advantage of the path diversity offered by N available APs, a client needs to install N radio devices, where N can be as large as the number of orthogonal channels offered by the WLAN (twelve for 802.11a), and 2) because multiple radios consume more energy, it may not be suitable for battery-powered clients.

Our approach for reducing the path switching cost is to use one radio on the client and deploy additional *secondary* access points in the WLAN infrastructure. In the extended MRD-TD architecture shown in Figure 4-8(b), each *primary* access point (PAP) defines a distinct WLAN cell and may be assigned a frequency channel that is orthogonal to a neighboring primary AP. The primary AP handles *authentication* and *association* procedures to allow clients to join its cell. Then, one or more secondary APs are placed within a cell, *i.e.,* within the coverage area of their primary AP, but at spatially diverse locations to achieve path diversity gains. The secondary APs are used to provide alternate transmission paths to the clients within their cell. All APs within a cell operate in the same frequency channel to minimize the path switching cost among them. A CM runs at each AP within a cell to monitor the wireless link condition and report feedback to the TDS, as described earlier. The TDS performs fine-grained path selection as previously described, except that the set of possible alternate paths for a particular client is limited to the APs within the client's cell.

Because common WLAN systems (*e.g.,* 802.11) offer link layer services (*e.g.,* security) that process packets in an AP, it is convenient to place the TDS inside the primary AP. Doing so allows the TDS to forward data frames to an AP *after* they have been processed. The existing wireless services should run without modification. More importantly, the processing takes place in a central location for every cell. Central processing obviates the need to distribute state across the secondary APs to run the existing services.

Although a secondary AP is similar to a primary AP, it is different in the following ways. A primary AP defines a distinct cell and a pair of neighboring primary APs define a cell boundary. Although a secondary AP may transmit frames within a cell, a secondary AP does not increase the size of the cell defined by its primary AP (when the operators deploy secondary APs within their primary AP's cell boundary). The primary AP is the only AP within a cell that handles *authentication* and *association* procedures to allow clients to join its cell. Thus, a secondary APs has no effect on a client's handoff policy, which dictates when a client initiates a handoff as it crosses cell boundaries. We made these design choices to eliminate the potential interference problems described in Section 4.2.3.

A significant advantage of the extended MRD-TD architecture is that it allows the WLAN to increase its capacity using the well-tested cellular architecture, while facilitat-

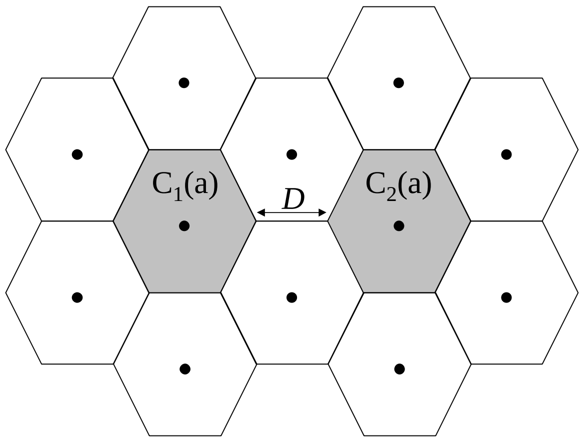


Figure 4-9: A hexagonal cell model. The cells C_1 and C_2 operate in channel a and may cause co-channel interference between each other. If all cell sizes are identical, the worst-case co-channel interference between C_1 and C_2 is a function of their minimum separation distance D.

ing low-cost fine-grained path selection. Because access points are commodity devices, we expect that secondary APs will not significantly increase deployment cost. If the cost of installing and wiring secondary APs at different physical sites become significant, WLAN operators may co-locate the secondary APs of one cell with the primary APs in the neighboring cells (see Figure 4-8(c)). In this case, MRD-TD operates in the same way as before except that alternate paths of a cell will extend into the neighboring cells. Consequently, the channel quality of the alternate paths may decrease and co-channel interference between cells that operate at the same channel may increase. In practice, however, the channel quality of alternate paths and co-channel interference between cells are related to the individual cell's location and traffic load. While MRD-TD does not restrict where secondary APs are deployed, a WLAN operator needs to make the appropriate trade-offs between cost and performance when deploying MRD-TD. The next section describes how MRD-TD, with strategic placement of secondary APs, limits the potential increase in co-channel interference.

4.2.3 Co-channel Interference

Co-channel interference arises when two or more wireless devices that operate at the same frequency are placed near each other (*i.e.,* within each others' radio interference range). Thus, APs that operate in the same frequency channel should be placed carefully so that they do not interfere too much with one another and reduce the overall capacity of the network. Because the extended MRD-TD architecture requires secondary APs that operate at the same frequency, it is important to understand how their deployment might affect the overall capacity of the network.

We use a simple hexagonal cell model as shown in Figure 4-9 to examine the impact of adding secondary APs into a WLAN. To greatly simplify our exposition, we assume that an AP is located in the center of each cell and that all cells are the same size. The results are general and remain valid even if the cell sizes and shapes are different. Without loss of generality, assume that cells C_1 and C_2 operate in the same frequency. In a traditional

WLAN without secondary APs, clients can move to the edge of a cell's boundary. Thus, the *worst-case* co-channel interference between the two cells is a function of the cells' minimum separation distance D, *i.e.,* the minimum distance between a client from C_1 and a client from C_2.

Suppose the primary AP is located at the center of every cell, adding secondary APs *does not increase* the worst-case co-channel interference, if the following conditions hold:

- The secondary APs do not affect a client's handoff policy. As a client crosses a cell boundary, it disassociates with the primary AP of the cell that it is leaving and associates with the primary AP of the cell that it is entering.
- Secondary APs are always placed within the boundary of the same cell as their primary AP.

The first condition maintains that a client cannot join a cell, *e.g.,* C_1, unless it is within C_1's cell boundary. Thus, regardless of the secondary APs' existence, all of C_1's active clients are still contained within C_1's boundary, and similarly for clients in C_2. The second condition ensures that the secondary APs of C_1 are placed within C_1's boundary, and the secondary APs of C_2 in C_2's boundary. With these two conditions, there is no way to place a wireless station from C_1 at less than D length away from the closest wireless station in C_2. Thus, the worst-case co-channel interference remains unchanged.

4.2.4 Rate Adaptation

When all the available transmission paths suffer from high long-term frame loss rates, fine-grained path selection could be used to avoid a substantial number of frame losses but the overall frame loss rate is likely to remain high. To help cope with channel conditions where transmit diversity becomes inadequate, we can adjust the transmission bit-rate using the rate adaptation algorithm listed in Figure 3-5.

Because there are multiple transmitters in a MRD-TD system, there is an interesting design choice of 1) running a single, global instance of the rate adaptation algorithm in the TDS to select a bit-rate based on the link layer losses observed by all transmitters or 2) running a separate, independent instance of the algorithm at each transmitter. How fine-grained path selection benefits the "global" approach of adapting bit-rates is clear. As we will show later, our system helps reduce the overall frame losses and the variance of frame losses. Both of these properties can help the rate adaptation algorithm select higher bit-rates to improve throughput, as we saw in the previous chapter. However, in the "distributed" rate adaptation scheme, it is unclear how fine-grained path selection will interact with the independent rate adjustments at each transmitter.

In this chapter, we provide an evaluation of MRD-TD using a fixed bit-rate transmission scheme and in the next chapter, using a global rate adaptation algorithm. The study of different rate adaptation schemes and their interactions with the MRD-TD sub-system is a rich topic for future work.

4.3 802.11 Implementation

We use Linux PCs equipped with a Intersil Prism-II based 802.11b PCI card to implement the primary and secondary APs, and a dedicated 100 Mbps Ethernet to serve as the wired backbone between the primary and secondary APs of a single MRD-TD WLAN cell. We modify the HostAP (ver. 0.0.1) driver [50] to incorporate the TDS and CM. We configure the wireless interfaces to run in 802.11 AP mode, and our prototype implementation works with regular, unmodified 802.11b managed mode clients.

4.3.1 Routing

We configure the wired backbone and the wireless network as different subnets. The AP host uses Linux *iptables* to forward packets with an IP address destined to a WLAN client from the wired network to the wireless network. Because we implement the TDS and CM within the HostAP driver, we need a way to deliver MRD-TD control messages to a TDS or CM running at a remote AP. We achieve this behavior by configuring each AP host with an IP address in the wireless subnet. An AP sends a MRD-TD control message to the wireless IP address of the target AP host via the wired backbone. Unfortunately, IP packets that reach the destination host will be consumed by the host. Thus, the target MRD-TD component running within the wireless interface's driver will not receive the control packet. To solve this problem, we add to every AP host one static ARP entry that contains the wireless IP address of the corresponding AP host. As long as there is an ARP entry with the AP's wireless IP address, the target host will forward all IP packets to the wireless interface, independent of the MAC address value in the ARP entry.

We implement the TDS inside the data path of the HostAP driver so that the primary AP can forward a client's packets to the secondary APs via the wired backbone. The TDS contains a table of wired ethernet MAC addresses of all the secondary APs in the primary AP's cell. Before the TDS forwards a packet to a secondary AP via the wired interface, it changes the packet's destination MAC address to the Ethernet address of the selected secondary AP listed in the table. We disable packet retransmissions in the native wireless interface layer to allow the TDS to assume control of retransmissions. The MRD-TD testbed used to collect experimental results in this chapter does not implement TDS-controlled retransmissions; the wireless interface simply drops all packets that fail their first transmission attempt.

A TDS also needs to determine which APs are within transmission range for a particular client. For the primary AP, client detection occurs automatically when the client sends an association request. Secondary APs currently do not detect whether a client is within range. However, we can implement client detection by configuring a dedicated wireless interface in every secondary AP to sniff for the client's upstream transmissions (either a data or an ACK frame). Similar techniques have recently been proposed to build connectivity graphs to improve 802.11 handoff performance [82].

4.3.2 Channel Monitor

We implement the CM inside the HostAP driver as an interrupt handler, which receives a callback triggered by a packet transmission, indicating if its delivery has succeeded or failed. The CM runs the MRD-TD path-selection heuristic. aWhen the CM detects that an AP's channel condition has fallen into a bad state, it sends a path-condition update to the TDS. The TDS then selects a different AP by cycling through the table of secondary APs. The TDS sends a control message to clear the packet history of the CM running at the selected secondary AP and to start forwarding subsequent packets to it.

4.3.3 Primary and Secondary Access Points

The primary AP runs like an ordinary 802.11 access point. It broadcasts periodic beacon messages to advertise its existence to clients and accepts their association requests. The secondary AP receives and forwards packets over the wireless interface, but does not participate in broadcasting beacons or associating with clients. We change the MAC address of the secondary AP's wireless interface to the MAC address of the primary AP's wireless interface. Hence, the secondary AP is configured to spoof the primary AP's identity. The MAC address spoofing allows a client in 802.11b managed mode to receive packets from different APs transparently and without interrupting the data flow.

One important detail concerns the use of link layer sequence numbers in 802.11. Ideally, the primary and secondary APs should use synchronized sequence numbers so that the packet's origination is completely indistinguishable. However, the Prism-II chipset used in our implementation does not export an API that allows us to synchronize the sequence numbers or to modify them in the 802.11 header. In practice, the sequence numbers are used only for duplicate packet detection and reassembling fragmented data frames. As long as we restrict the fragmented frames to the same wireless interface that initiated the link layer fragmentation, the system will handle fragmented frames properly.

Unfortunately, the system can no longer detect duplicate data frames transmitted by different APs, which can happen in the event of an ACK frame loss.[3] Fortunately, link layer packet duplication is usually not a problem in practice because the best-effort service model allows for occasional packet duplication; the link layer is not required to filter all duplicate packets for the higher layers of the protocol stack. End-host applications and transport layer protocols such as TCP can usually detect duplicate packets and discard them if necessary.

We have not yet implemented the client-side modifications to support fine-grained path selection in the uplink direction. Uplink transmissions from the unmodified clients are received and acknowledged by the primary AP in exactly the same way as a regular AP in the 802.11 network. We also have not implemented the TDR, which runs on the receiver and is used to order packet delivery for higher transport layers. In the next chapter, we will show how the RDC can be used in place of TDR to ensure in-order packet delivery in the integrated system.

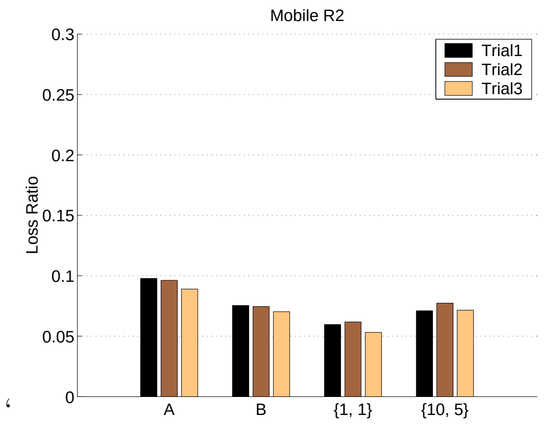
4.3.4 Security Support

MRD-TD does not affect link layer security services such as the Wired Equivalent Privacy (WEP) and the 802.1x security extensions [17]. An unmodified client associates and authenticates with a primary AP in the same manner as it would in the original 802.11 WLAN. For downlink communication, the TDS can let the WEP/802.1x security service perform the necessary processing to a data frame before forwarding it to the selected AP for immediate transmission. Since the security layer is typically implemented in the device driver of the wireless interface, it is important to run the TDS component inside the primary AP.

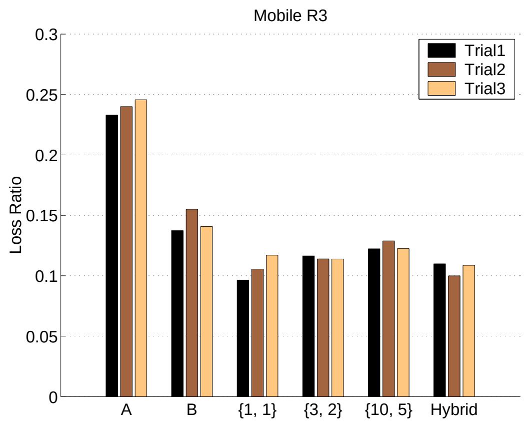
For clients that do fine-grained path selection in the uplink direction, the TDS at the client may modify the next-hop address in the 802.11 link layer header of an uplink frame. Since the 802.11 header is not protected by either WEP or 802.1x, the security service

[3] In the future, the availability of an 802.11 chipset that allows higher-layer control of the frame sequence numbers can solve this problem in a way that permits duplicate detection.

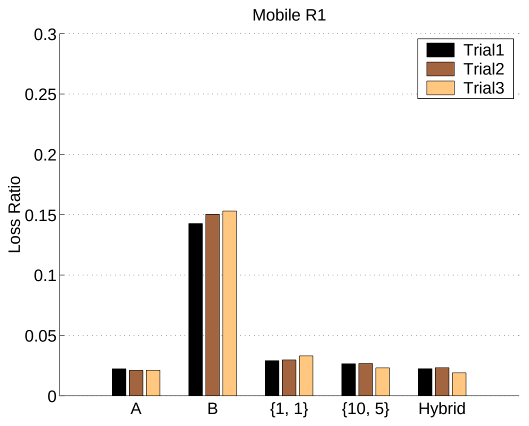
for uplink packets should be unaffected, as long as the secondary APs forward all received packets to the primary AP for proper processing.



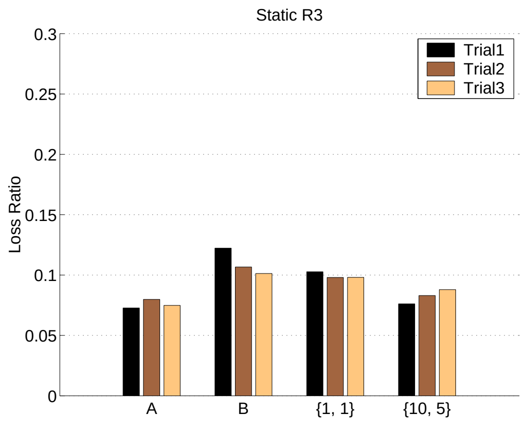
(a) Mobile at R_2



(b) Mobile at R_3



(c) Mobile at R_1



(d) Static at R_3

Figure 4-10: Average frame loss rates of different transmission schemes at three different receiver positions. The label denotes that values $\{H, T\}$ used in a MRD-TD transmission scheme. Hybrid uses $\{1, 1\}$ for transmitter A and $\{3, 2\}$ for B.

4.4 Experimental Results

We evaluate our implementation of MRD-TD to demonstrate the benefits of fine-grained path selection. The experimental setup is the similar to the one in Section 4.1.1. The major difference is the use of unicast frames as opposed to broadcast frames, and the transmitters at A and B are APs running our MRD-TD implementation. To measure link layer frame loss rate (FLR), we disabled packet retransmissions. We stream 1500 byte unicast UDP packets to the receiver at each of the three locations in Figure 4-1 at a rate of 240 packets per second using i) only transmitter A (referred to as scheme A), ii) only transmitter B (referred to as scheme B) and iii) MRD-TD with several settings of H and T values. As explained earlier, MRD-TD will use AP A or B to transmit each frame. Except for the *Hybrid* configuration, the same set of H and T values is used as the switching criteria from

A to B and from B to A. Under the *Hybrid* configuration, MRD-TD uses $H = 1$ and $T = 1$ as the switching criteria from the transmitter with a higher average FLR and $H = 3$ and $T = 2$, from the transmitter with a lower average FLR.

We disabled roaming at the receiver to prevent it from initiating a handoff during the experiment. We conducted our experiments in late evening to avoid interacting with the building's daily 802.11 activity. We repeated each experiment over three trials. To avoid biases from the human subject performing the mobile experiments, the order of the experiments' trials was randomized and was unknown to the human subject. Each trial transmitted 72,000 packets in 300 seconds.

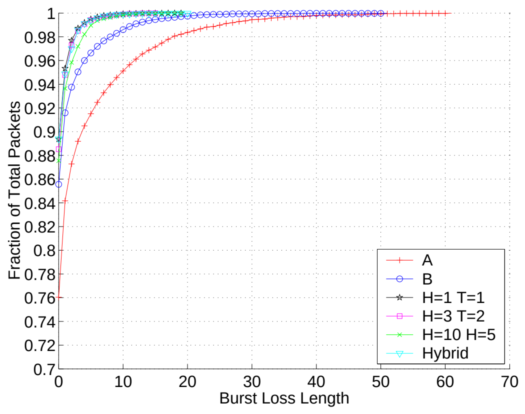
These experiments are by no means exhaustive. Nonetheless, they illustrate how a simple fine-grained path selection heuristic such as the one used by MRD-TD can offer significant performance improvements in terms of reduced delay and loss rate in realistic scenarios.

4.4.1 Frame Loss Rate

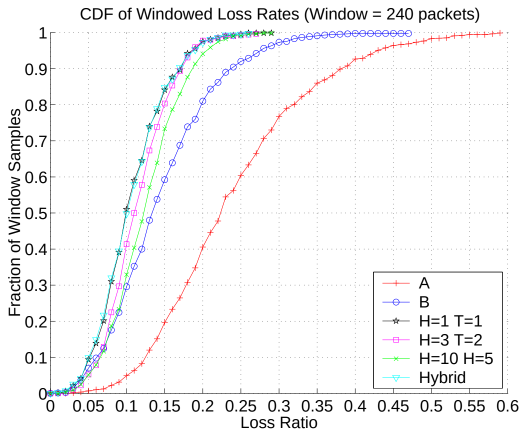
In our mobile environment, MRD-TD performs significantly better than both schemes A and B when the receiver is at R_2 and R_3, for all the values we used for H and T. Figures 4-10(a) and 4-10(b) show that at R_2, MRD-TD $H = 1$ and $T = 1$ reduces the average FLR by about 38% from scheme A and 21% from scheme B, and as the receiver moves further to R_3, the loss reductions increase to 56% from A and 26% from B. As predicted in Section 4.1 (see Figure 4-3), MRD-TD effectively reduces losses by avoiding burst losses in the wireless channel. MRD-TD performs better with $H = 1$ and $T = 1$ than with $H = 10$ and $T = 5$ because it is more responsive with smaller H and T values.

At R_1, the receiver is much closer to transmitter A than B. As expected, the average FLR of scheme A (2.1%) is much lower than B (15%). Due to the large difference in the average loss rates, it is unlikely that the auto-conditional loss probability of transmitter A ($P(A_{i+k}|A_i)$) will exceed the cross-conditional loss probability of B ($P(B_{i+k}|A_i)$) for any lag k. Figure 4-10(c) shows that non-*Hybrid* configurations of MRD-TD perform slightly worse than schemes A and B. To compensate for the large differences in the average loss rates between the two transmitters, MRD-TD may use different H and T path-switching thresholds for each path. The *Hybrid* case at R_1 uses a more conservative threshold (*i.e.,* $H = 3$ and $T = 2$) for the transmitter with lower FLR (A), and maintains an aggressive threshold (*i.e.,* $H = 1$ and $T = 1$) for the transmitter with higher FLR (B). Figure 4-10(c) shows that *Hybrid* MRD-TD performs as well as the better transmitter (A) when it uses different path-switching thresholds for different paths. Thus, MRD-TD can adapt remarkably well to extremely asymmetric, dynamic channel conditions (*e.g.,* R_1) by making small adjustments to H and T. In our mobile environment, MRD-TD performs no worse than the best available path when the available paths are extremely different (*e.g.,* R_1). But when the paths are less asymmetric (*e.g.,* R_2 and R_3), MRD-TD drastically reduces the average FLR compared to the fixed-path schemes.

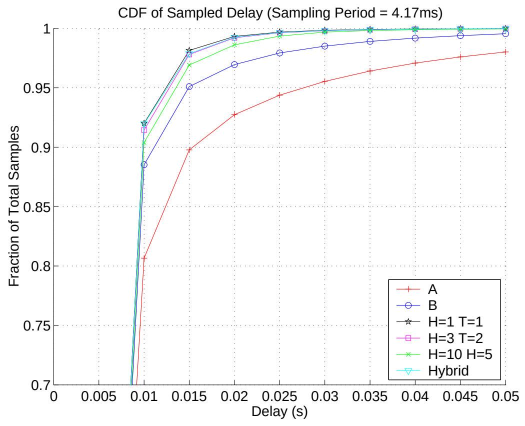
When the receiver is stationary at R_3, Figure 4-10(d) shows that MRD-TD has a lower average FLR than scheme B but a higher average FLR than scheme A. This is because losses are seldomly bursty in our static environment. In our case, choosing the transmitter that has a lower average FLR (which is AP A) is better than fine-grained selection. However, we believe that there are cases when fine-grained path selection is beneficial even when the receiver is static, *e.g.,* when the channel condition is dynamic or when there is localized interference at the transmitter.



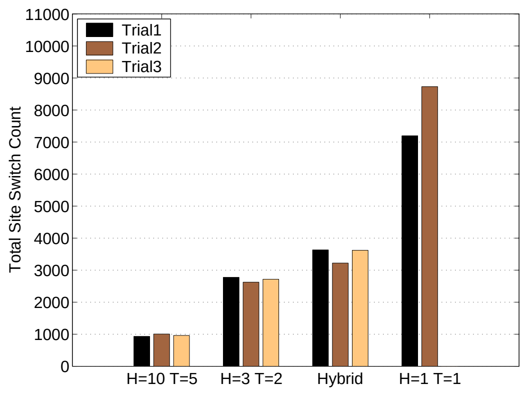
(a) Burst loss length



(b) 1-second window loss rate



(c) Sampled channel delay



(d) Number of site switches

Figure 4-11: CDFs of various measures for the mobile experiments at R_3

The measured FLR suggests that the performance gains of MRD-TD is much higher in dynamic conditions than in static conditions. To understand MRD-TD's potential gains in other performance aspects, we focus on mobile experiments at R_3 for the rest of this section. In general, the trends described earlier hold for the following evaluation: MRD-TD performs no worse than the best available path at R_1. When the receiver is stationary at R_3, MRD-TD's performance is about the same as the average of the two available paths.

4.4.2 Burst Loss Length and Window Loss Rate

Figure 4-11(a) shows the CDF of the burst loss length. MRD-TD is able to significantly cut the tail of the distribution. In particular, the largest burst loss length of MRD-TD with $H = 1$ and $T = 1$ is less than 20 whereas that of scheme A is 52 and that of scheme B is 61.

Some applications such as multicast video streaming require low loss rates over short intervals. Figure 4-11(b) shows the CDF of frame loss rates over 1-second windows. MRD-TD's distribution of the 1-second window FLRs is much lower (and narrower) than that of both schemes A and B. The worst-case 1-second window loss rate is also much lower: the highest loss rate in a 1-second window for schemes A, B and MRD-TD are 59%, 47%, and 29% respectively.

4.4.3 Channel Delay

As explained earlier, losses tend to occur in bursts, especially in mobile environments. Transmitters often experience periods of degraded channel conditions, lasting for several tens of milliseconds. Any frame transmission attempt during such periods is likely to fail. We define the *per-packet channel delay* as the difference between the time when a packet is first transmitted in the wireless medium and the time when it is successfully received. Channel delay is an important metric for voice and video applications that require low one-way packet delay and low delay jitter.

To accurately compute the per-packet channel delay, we need the transmit and receive times of each packet. It is difficult to synchronize the clocks accurately to within a fraction of a microsecond and consistently among the transmitters and the mobile receiver in each experiment. Thus, we use a sampling approximation to estimate per-packet channel delay. We transmit packets at periodic intervals of I and assume that packet i is sent precisely at $t_i = i * I$. Let r_j be the time when the j^{th} data frame is successfully received, *i.e.,* when the jth data frame logged in the receiver's data trace. Then, the i^{th} channel delay sample d_i is computed as: $d_i = r_j^{min} - t_i = r_j^{min} - i * I$, where r_j^{min} is the minimum r_j in the data trace that satisfies $r_j - t_i \geq I$. In our analysis, we combine all three trials of each experiment to generate $216,000$ delay samples. We used an inter-packet transmission period of $I = 4.17$ ms.

Figure 4-11(c) shows the CDF of the channel delay samples for the mobile experiment under various schemes when the receiver is at R_3. As shown, all of the MRD-TD schemes have a lower channel delay distribution than the fixed-path schemes (A and B), *e.g.,* in MRD-TD, 98% of the packets have a channel delay less than 15 ms but in the fixed-path schemes A and B, fewer packets (90% and 95%) are transmitted successfully within the same delay. In terms of delay reduction, the 99*th*-percentile delay is reduced from 70 ms and 40 ms for fixed-path schemes A and B to 20 ms for MRD-TD.

4.4.4 Number of Path Switches

We measured the number of path switches that took place in each of our experiments. A large number of switches is indicative of a large number of MRD-TD control messages being sent over the wired backbone. Because a typical wired backbone usually has a much higher capacity than a 802.11b WLAN, the additional traffic caused by MRD-TD's control messages in most cases will produce little congestion in the wired network. However, for low-bandwidth wired backbone networks, a smaller number of AP switches may be more desirable especially with higher-capacity WLAN technologies such as 802.11a.

Figure 4-11(d) shows the number of AP switches for various MRD-TD configurations. As expected, the number of switches decreases as H increases. The number of switches for the third trial of MRD-TD with $H = 1$ and $T = 1$ is missing because the file containing that data was corrupted.

4.5 Chapter Summary

In this chapter, we showed that a fine-grained path selection technique for wireless networks can yield substantial performance benefits under the following conditions: i) strong temporal loss correlation within a path in which the short-term (10-100 ms) frame loss rate is significantly higher than the steady state frame loss rate, and ii) weak spatial loss correlation across paths. Using a number of real-world experiments on an indoor 802.11b WLAN, we showed that such conditions can occur when the receiver is in motion. Our results show that the simple and practical fine-grained path selection technique proposed in this paper can help reduce loss rates—without consuming extra wireless bandwidth—by as much as 26% compared to a fixed-path scheme that uses the best available transmission path under realistic settings.

Our choice of fixed algorithm parameters (loss history and loss threshold) for fine-grained selection may not be appropriate in some environments. An interesting direction for future work is to explore adaptive path selection algorithms so that the scheme is suitable under a variety of dynamic conditions.

Chapter 5

MRD-TRD: Integrating Transmit and Receive Diversity

This chapter describes how to integrate the MRD-RD and MRD-TD sub-systems to provide both transmit and receive diversity gains in a unified system called MRD—Transmit/Receive Diversity (MRD-TRD). One way to integrate the two sub-systems is to run them independently in the opposite directions of communication. For example, a MRD WLAN can run MRD-RD to perform frame combining for uplink traffic and MRD-TD to perform fine-grained path selection among multiple APs for downlink traffic. This design takes advantage of path diversity provided by multiple APs in the infrastructure and supports clients that have only a single radio.

For clients that have multiple wireless interfaces, we can run both MRD-RD and MRD-TD to provide both receive and transmit diversity gains in the same direction of traffic. For example, in the downlink direction, MRD can provide receive diversity gains using multiple receive radios installed on the client and provide transmit diversity gains from multiple APs in the infrastructure. Intuitively, the gains of the integrated system are expected to be greater than the individual transmit or receive diversity systems because the integrated system provides additional communication paths between the client and the infrastructure. Transmit diversity may be used to divert transmissions away from physical obstacles, while receive diversity may be used in the same direction of communication to mitigate the effects of attenuation, mobility, multipath, and thermal noise in the electronics of the receiver radio.

Chapters3 and 4 showed how each sub-system implements its own feedback-control logic. MRD-RD uses the RDS and RDC to run the RFA protocol, while MRD-TD uses the TDS and TDR to perform path selection and packet ordering. In merging the two sub-systems for delivering packets in the same traffic direction, we have to decide how to connect the data and control flows (*i.e.,* retransmissions, rate adaptation, packet ordering) between the RDS and TDS components at the sender and between the RDC and TDR at the receiver.

Sections 5.1 and 5.2 describe the design and implementation of the MRD-TRD system. We evaluate the performance of MRD-TRD in Section 5.3 and summarize the chapter in Section 5.4. Our experiments show that the combined MRD-TRD system provides up to 34% and 12% throughput gains over schemes using MRD-TD and MRD-RD respectively.

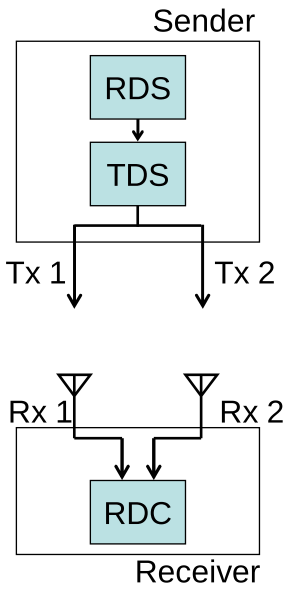


Figure 5-1: An illustration of the MRD-TRD architecture, which combines both transmit radios Tx 1 and Tx 2 into a single virtual MRD-TD link and runs MRD-RD over this link.

5.1 Design of MRD-TRD

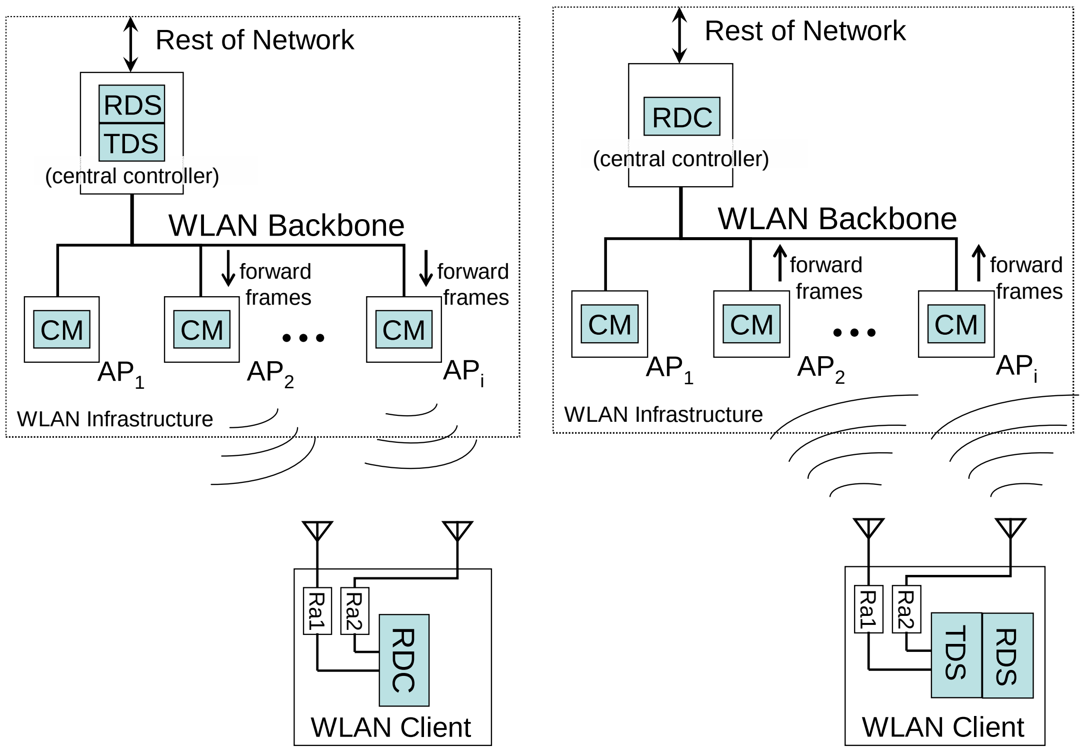
MRD-TRD uses MRD-TD to tie different transmit radios into a single virtual MRD-TD link and runs MRD-RD on top of the virtual link, as depicted in Figure 5.1. In this configuration, the higher network layers inject packets into the RDS of the MRD-RD sub-system, which encapsulates them into link layer frames and passes them to the TDS of the MRD-TD sub-system. In turn, the TDS performs fine-grained path selection directly over a set of transmit radios. In the original MRD-TD design (see Section 4.2.1), each CM running at each available transmitter keeps a history of transmission losses and notifies TDS to switch paths if the frame loss rate exceeds a threshold. In MRD-TRD, we modify the CM to report to the TDS whether it has received a link layer ACK for every frame it transmits. The TDS assumes from the CM the task of tracking link layer frame losses for each transmission path, and performs fine-grained path selection accordingly.[1] The TDS also forwards the link layer ACK reception reports to the RDS so that the RDS can issue an RFA to the RDC quickly after a transmission loss (detected by the absence of an ACK) at the link layer.

In MRD-TRD, only the MRD-RD layer knows about the final status of each frame transmission at the receiver. Thus, we disable the TDS from retransmitting frames that do not receive a link layer ACK. Otherwise, the TDS can potentially cause unnecessary retransmissions for frames successfully received by the passive radio at the receiver or successfully combined after the RDC stage at the receiver.

The receiver of MRD-TRD is identical to the one in the MRD-RD sub-system. Although the sender runs the TDS, MRD-TRD does not run the corresponding TDR component at the receiver because the RDC already implements an ordering buffer that the TDR implements.

Figures 5-2(a) and 5-2(b) illustrate how each of the MRD-TRD components fit together

[1]The TDS can also cross the layer boundary between MRD-TD and MRD-RD, and use the feedback of MRD-ACKs in conjunction with the loss status of link layer transmissions to perform path selection. We leave the development of this idea for future work.



(a) Downlink (b) Uplink

Figure 5-2: WLAN architecture of MRD-TRD.

to deliver downlink and uplink traffic in a single WLAN cell. In the downlink direction, the central controller provides transmit diversity and performs fine-grained path selection among the available APs (tuned to operate in the same radio frequency), with one of the APs (*e.g.*, AP$_1$) configured as the primary AP and the rest as secondary APs.[2] The primary AP broadcasts beacon frames that the WLAN clients use to measure signal strength and determine which cell to associate with. The secondary APs spoofs the identity of primary AP so that path switches can take place transparently from the WLAN client. At the client, the RDC collects received frames from multiple radios. Ra1 is the active radio that responds to any successful link layer receptions with a link layer ACK, while Ra2 is tuned to listen on the same radio frequency as Ra1 and provides an alternate reception path for receive diversity at the client.

In the uplink direction, the radios reverse their roles. The client uses radios Ra1 and Ra2 (without distinction) for transmit diversity. The set of APs provide receive diversity for uplink traffic, with one of the APs configured as the active AP that transmits link layer ACKs in response to successful link layer receptions.

5.1.1 Rate Adaptation

Our MRD-TRD architecture supports two forms of rate adaptation. In our implementation, the RDS runs a single, global instance of the rate adaptation process and adjusts bit-rates

[2]As described in Section 4.2.2, the secondary APs may be provided by adding APs within a cell or by co-locating them at the sites of primary APs that belong to adjacent cells

according to the loss rates observed by the RDC. Alternatively, we can run a separate instance of the rate adaptation process for each transmit radio and have each of them select a transmission bit-rate independently. Thus, similar to MRD-TD's "distributed" rate adaptation scheme described in Section 4.2.4, each transmit radio is associated with its own rate adaptation process. Unlike MRD-TD's distributed scheme, which adapts to link layer losses, MRD-TRD's distributed scheme adapts to RDC loss statistics for improved efficiency. Thus, the distributed rate adaptation processes need to peek into the MRD-ACKs for the final frame loss status that the RDS receives from the RDC.

Because a rate adaptation process should only adapt to the reception status of frames transmitted by the radio that it controls the bit-rate, each rate adaptation process needs to maintain a set of sequence numbers of the frames transmitted by its transmit radio. The rate adaptation process use the saved sequence numbers to deduce which loss status bits in a MRD-ACK are relevant to it and adapt bit-rates accordingly.

We expect the "global" MRD-TRD rate adaptation scheme to behave similarly to the one used in the MRD-RD sub-system (Section 3.4), except that it draws further benefits from reduced frame losses provided by fine-grained path selection. Although running an independent rate adaptation process for each transmissions path could in theory increase adaptability, the distributed rate adaptation scheme presents many complex interactions with both the TDS and the RDS components. It is unclear how these interactions might impact performance. For instance, one of the distributed rate adaptation processes might select a low bit-rate for its transmit radio A, while another process selects a high bit-rate for its transmit radio B. It is plausible that B can observe both higher link layer frame loss rates and higher throughput than A (*e.g.,* the MRD-RD sub-system can recover most of the link layer frame losses in B but not for A). Despite the higher throughput in path B, TDS selects A for transmission because it observes fewer link layer frame losses (perhaps because the rate adaptation algorithm selects a low bit-rate for A). One possible solution is to modify the fine-grained path selection algorithm to include throughput as part of its path selection metric. Developing a robust and stable distributed rate adaptation scheme for MRD-TRD is beyond the scope of this dissertation and we defer its study to future work.

5.2 Implementation

We ported the MRD-TD sub-system from the HostAP/Prism2 802.11b platform to the MADWiFi/Atheros 802.11a/b/g platform on which the MRD-RD sub-system is implemented. Rather than integrating the TDS logic inside the wireless interface's driver as we did in our old implementation, we decided to implement the TDS as a separate user-level program called `mrdsender`. We also ported the RDS logic to `mrdsender` to facilitate the integration of the RDS and TDS components in the MRD-TRD system. The clean separation of the sender control logic from the driver allows us to run `mrdsender` at the client or on a router attached to the WLAN's backbone infrastructure. We can easily set the `mrdsender` and the RDC daemon to communicate in different schemes (single-radio, MRD-TD, MRD-RD, MRD-TRD).

Because we separated the RDS and TDS implementations from the driver, the `mrdsender` uses UDP to transmit and receive frames between it and the APs or diversity radios. The `mrdsender` encapsulates all transmission frames along with their transmission bit-rate and transmit sequence numbers (explained later) into UDP datagrams. The `mrdsender` sends the datagrams to a target `mrdforwarder` daemon over the backbone network (or via the loopback interface in the case when both `mrdsender` and `mrdforwarder` run on the same host, *e.g.,* a WLAN client). There is one `mrdforwarder` daemon process running per terminal. Upon receiving the datagrams, the target `mrdforwarder` decapsulates them and forwards the resulting transmission frames and their specified bit-rates into the intended wireless interface's raw socket for immediate transmission at the specified bit-rates. We modified the MADWiFi driver to accept and select the bit-rate specified for each frame and return the status of the transmissions (*i.e.,* the success or failure of receiving the link layer ACK for the transmitted frame) to the `mrdforwarder`, which then relays it to the `mrdsender`.

In addition to the changes above, we introduced four optimizations in MRD-TRD to improve its efficiency. We describe their details below.

5.2.1 Improving throughput: Pipelined transmissions

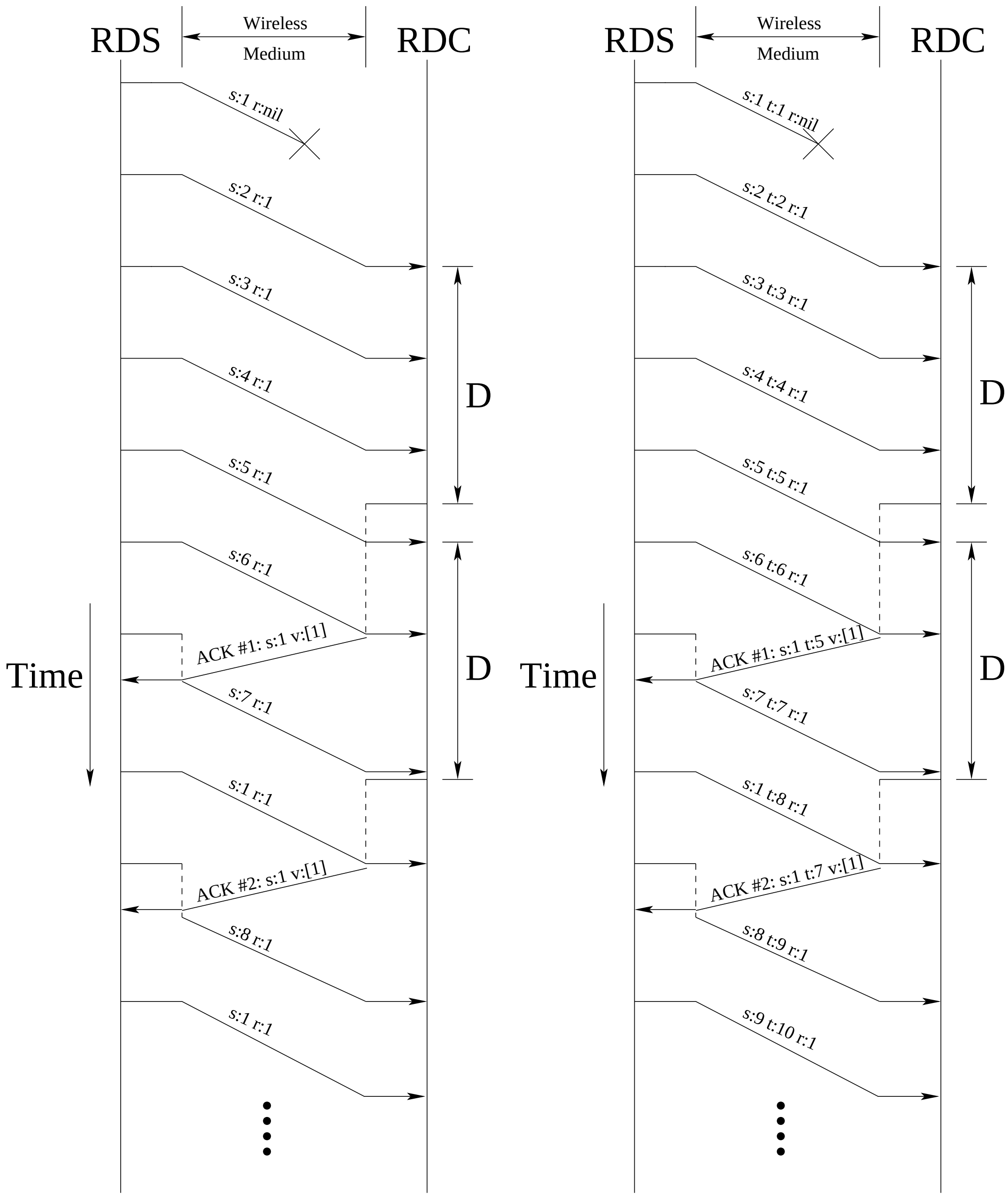
The delay for the `mrdsender` to receive a feedback after a transmission takes at least several hundred microseconds, which is comparable to the duration of transmitting a frame. To improve throughput, the `mrdsender` does not wait for the status of every frame transmission before transmitting the next frame, *i.e.,* the transmissions are pipelined. In our implementation, the depth of the pipeline, P, is 4 frames and appears to work well on our testbed. The `mrdsender` blocks if more than P frames are transmitted without receiving their corresponding feedback from the `mrdforwarder`.

One side effect of pipelined transmissions is that it can increase retransmission delays. The original implementation assumes that RDS receives the transmission status of a frame immediately after its transmission so that it can set the *rfa* flag with the appropriate value in the CTX header (Figure 3-7(a)) of a subsequent frame. Because the `mrdsender` pipelines the transmissions, the RDS might not receive the feedback of a transmitted frame until after sending one or several subsequent frame(s) into the pipeline. Hence, the RDS may not issue an RFA for a failed transmission until after (up to) P frames have been transmitted. The RFA delays add delays to obtaining MRD-ACKs from the RDC, which in turn add delays to retransmissions. However, the increased delays are usually insignificant ($\approx 1-2$ ms) because P is small (4). Also, the RFA delays should not affect throughput because of pipelined transmissions.

After implementing pipelined transmissions and conducting initial tests, we observed increased levels of link layer losses in the MRD-TD and MRD-TRD schemes. Because of pipelining, the `mrdsender` can queue up (up to) $P-1$ transmission frames behind a transmit radio before the TDS switches subsequent transmissions to a different but nearby transmit radio. Now, multiple transmit radios have backlogged frames for transmission, thus increasing the level of contention and collisions in the channel.[3] To alleviate contention, we incorporated a mechanism in the `mrdsender` to let the pipeline drain before the TDS is allowed to switch transmission paths. Although the system sacrifices some throughput on every path switch, the benefits of fine-grained path selection still outweigh its costs, as we will see in Section 5.3.

[3]The minimum contention window size in 802.11a and 802.11g has reduced from 31 slots (in 802.11b) to 15 slots. Thus, there is $\frac{1}{15} = 6.67\%$ chance for a collision in a scenario that involves two backlogged transmit radios.

5.2.2 Identifying stale feedback from the RDC: Numbering the transmission sequence



(a) Problem (b) Solution

Figure 5-3: Time diagram showing the problem of stale feedback (left) and its solution (right). Symbols s, t, r, v label respectively the values for *seq*, *tseq*, *useq* (for RFA), and the MRD-ACK status feedback vector. Note that only one transmission can occupy the channel in the wireless medium at a time. Dotted lines indicate the length of time that a frame resides inside the transmit radio's buffer.

We discovered that there were many instances where stale feedback from MRD-ACKs triggered unnecessary retransmissions. Figure 5-3(a) shows an example illustrating how it might happen. The example assumes no transmission pipelining and begins with the loss of frame 1 followed by a successful transmission of frame 2 that includes an RFA for frame 1. When the RDC receives frame 2, it sets the MRD-ACK timer D, which expires during the reception of frame 5. Then, the RDC reports the loss of frame 1 through MRD-ACK 1. Observe, however, that the final transmission status of frame 1 is unknown to the RDS thus far, so frame 5 includes a RFA for frame 1 and causes the RDC to set a fresh MRD-ACK timer to honor the RFA request.

Upon receiving MRD-ACK #1, the RDS learns about the loss of frame 1 and retransmits it. Meanwhile, the second MRD-ACK timer expires and the RDS sends MRD-ACK #2 before the arrival of frame 1's retransmission. Hence, MRD-ACK #2 contains stale information about the status of frame 1 because it reports the status of frame 1's first transmission (failure) as opposed to the status of its second transmission. Let's suppose that the second transmission of frame 1 fails to receive a link layer ACK but succeeds after frame combining. Because the RDS cannot tell whether the status in MRD-ACK 2 #is stale, it accepts the failure status of frame 1 from MRD-ACK #2 and transmits frame 1 for a third time, even though the RDC has successfully recovered the errors of frame 1's second transmission.

To solve the problem, we embed into each frame a 11-bit transmission sequence number, *tseq*,[4] which the RDS increments on every transmission, including retransmissions, independently from the frame's *seq* value (Section 3.5.2). Thus unlike *seq*, which remains fixed once assigned to a frame, *tseq* increases monotonically to allow the RDS compare the transmit order of any two given transmissions. Whenever the RDC sends a MRD-ACK, it informs the RDS about the latest transmission it has received thus far, by inserting the latest-received *tseq* value into the MRD-ACK header. The RDS can then decide that a given status bit in the MRD-ACK is stale if the current *tseq* of the frame in the retransmission buffer is greater than the *tseq* value indicated in the MRD-ACK. Figure 5-4 shows the new format of the data frame and MRD-ACK headers.

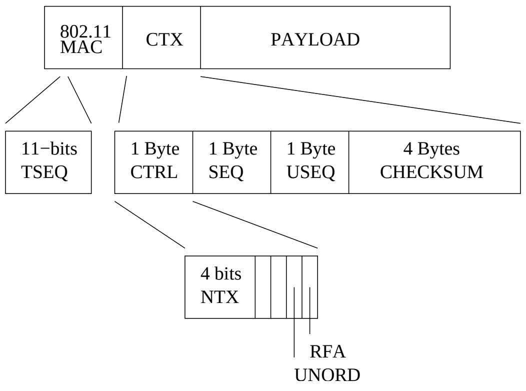
Figure 5-3(b) illustrates how *tseq* solves the stale feedback problem in our original example. Because the RDC receives the 7th transmission (frame 7) before sending MRD-ACK #2, the RDC assigns $tseq = 7$ in the MRD-ACK #2's header. Meanwhile, the second transmission of frame 1 is the 8th transmission in the sequence so $tseq = 8$. Because the latest transmission sequence of frame 1 is greater than the one found in MRD-ACK #2, the RDS considers frame 1's failure status in the MRD-ACK as stale and ignores it. Then, the RDS transmits the next frame (frame 9) in the sequence. The RDS will eventually receive the status of the second retransmission of frame 1 from MRD-ACK #3 (not shown).

5.2.3 Draining stalled traffic: Early retransmissions

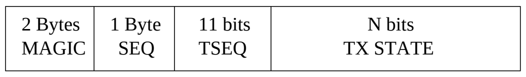
While running some TCP throughput tests over a low-loss link using MRD-RD or MRD-TRD in both traffic directions, we encountered large variations in the measured throughput between trials.

The main problem comes from our implementation of the RFA protocol, which uses *only* subsequent data frames to issue an RFA to the RDC. Thus, the RDS can fall into a bad state where it cannot issue an RFA because the transmission queue is empty. This

[4] We managed to use 802.11's 12-bit sequence number field to embed the 11-bit *tseq* field. The reason why *tseq* is 11-bits long is that it (conservatively) ensures that the *tseq* value do not encounter wrap-around value problems among the unacknowledged frames in the RDS retransmission buffer.



(a) CTX Header in the transmitted data frame



(b) MRD-ACK Packet

Figure 5-4: New MRD-ACK control information.

problem is present even when MRD-RD or MRD-TRD is running on one direction but is especially pronounced when a TCP connection runs over a system that has either MRD-RD or MRD-TRD enabled in both directions of traffic.

The following example illustrates how the problem usually happens. First, assume that the RDC on the sender side fails to receive one of the TCP ACKs as a result of frame errors. The RDS on the receiver side continues to transmit the subsequent TCP ACKs but may not issue an RFA because of feedback delays in the transmission pipeline (Section 5.2.1). The original frame loss presents a gap in the sequence so the RDC at the sender side keeps the incoming TCP ACKs in the ordering buffer. If, at the same time, the TCP window at the TCP sender is full, no additional forward TCP data packets would reach the TCP receiver and as a result, no additional TCP ACKs are generated. Thus, by the time that the link layer feedback reporting the loss of the original TCP ACK frame makes it back to the RDS in the TCP receiver, the receiver's transmission queue is empty, and it cannot issue an RFA (because RFAs were always piggybacked on data frames). In this case, both the TCP sender and TCP receiver are simultaneously waiting for timeouts to occur: The RDS on the receiver side cannot issue an RFA so it is waiting on a retransmission timeout ($T_s = 90$ ms). At the same time, the TCP sender is waiting for TCP ACKs, all of which are blocked inside the ordering buffer, and is waiting for a retransmission timeout ($RTO_{min} =$ 200 ms in Linux) before it retransmits the un-acknowledged TCP segment. As a result, the traffic stops dead in both directions for a substantial duration, impacting the throughput of the TCP flow.

The crux of the problem is that empty transmission queues are preventing the RDS from sending RFAs. One solution is to have the RDS detect when a transmission queue becomes empty and transmit a pure RFA frame as needed.[5] Alternatively, the RDS can retransmit one of the outstanding unacknowledged frames in the send buffer *early* (*i.e.,* without a

[5]Note that simply reducing the retransmission timer would not work because it can increase the number of unnecessary retransmissions even when the transmission queue is not empty.

MRD-ACK) whenever the transmission queue *and* the pipeline become empty. The RDS can use the early retransmissions to issue an RFA.[6] The latter strategy usually yields greater throughput because it does not need to wait for the MRD-ACK delay, D, before retransmitting frames. However, the cost of the latter strategy is the possibility of retransmitting frame that the RDC has received correctly. The RDS waits until the pipeline empties before retransmitting a frame early because we wish to perform early retransmissions during times when the channel is less busy. Although there is waste associated with retransmitting a successfully recovered frame, the cost of transmitting a pure RFA is there even if it is small. After weighing the trade-offs above, we chose to implement early retransmissions to drain stalled traffic to maximize throughput.

5.2.4 Out-of-order packet delivery

Because some applications can tolerate out-of-order packet delivery and it would be useful to flag them so that they can be immediately delivered to the higher network layers right after a successful reception, avoiding potential delays in MRD's packet ordering buffer. We have incorporated this feature in our current implementation of the MRD-RD and MRD-TRD systems. The new CTX header adds an *unord* bit field to flag frames for unordered, immediate transmission (Figure 5-4(a)).

Out-of-order packet delivery is especially useful for transmitting MRD-ACKs in a WLAN that has the MRD-RD or MRD-TRD system enabled in both directions of traffic. Because both RDC and RDS run on the same host machine, the RDC, after receiving an RFA from a remote terminal, can use the local RDS to transmit the MRD-ACK in the return direction. Doing so takes advantage of receive diversity in the return direction and improve the delivery rate of MRD-ACKs. Because MRD-ACKs do not require ordered delivery to the RDC, it is safe to mark them for immediate delivery. The RDC only inserts normal transmission frames into the ordering buffer so they would not block in the event of a lost MRD-ACK (or a loss of any other frames marked *unord*).

5.3 Evaluation

We run UDP throughput measurements to evaluate the performance of MRD-TRD. First, we describe our testbed setup followed by our throughput results. We present the underlying frame loss and recovery rates and the distribution of the selected bit-rates to help explain how the MRD communication schemes improve performance over the single-path schemes. We conclude the section by examining the packet delivery delay characteristics of the MRD-TRD system.

5.3.1 Setup

We conducted five UDP downlink experiments under highly varying channel conditions generated by a moving laptop client receiver. We used the same testbed setup as shown in Figure 3-8, except we configure node R1 as a primary (802.11a Master-mode) AP and node R2 as a secondary (802.11a Monitor-mode) AP.[7] We configure the laptop client C

[6]It may be odd at first to issue an RFA for the frame that is already being retransmitted. But often, there can be other unacknowledged frames in the send buffer and the returning MRD-ACK may contain useful status report about them.

[7]The host machines R1 and R2 have been upgraded to Intel Celeron desktops running at 2.66GHz.

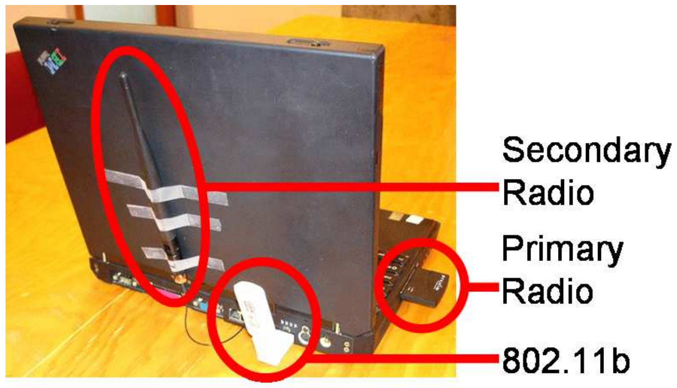


Figure 5-5: Laptop Setup. The primary radio is a Proxim 8480-WD 802.11a/b/g CardBus adapter and the secondary radio is an internal miniPCI module extracted from a Netgear WAG311 PCI card. The secondary is attached to an external 5dBi 2.4GHz omni-directional rubber duck antenna. The 802.11b USB dongle provides a back-channel connection to the building's production wireless network. The back-channel allows experiments to be launched and monitored from the IBM T30 laptop.

as a (802.11a Managed-mode) receiver. Our testbed also includes a central controller (not shown) that runs `mrdsender` and connects to R1 and R2 via a wired network to generate UDP traffic and run the sender-side processes in all of the experiments.

The five experiments evaluate the performance of the conventional schemes (R1 and R2) that used a single-radio AP, of the MRD-TD scheme (TD) that used multiple APs as transmitters, of the MRD-RD scheme (RD) that used multiple radios on the client as receivers, and of the MRD-TRD scheme (TRD). In the RD experiments, we enable only the downlink components of the MRD-RD sub-system (thus, the client uses multiple radios to receive frames but uses its primary radio to transmit MRD-ACKs via a single-path link in the uplink direction), while the TRD experiments enable uplink MRD-RD in addition to the downlink MRD-TRD components (thus, the client uses its primary to transmit MRD-ACKs via a MRD-RD link in the uplink direction). The reason why we enable uplink receive diversity in the TRD experiment is to keep the system's implementation simple. Our implementation of `mrdforwarder` automatically forwards frames to and from any operating radios. Because the TRD experiments enable the secondary AP for transmit diversity in the downlink direction, `mrdforwarder` will automatically forward uplink traffic to the `mrdsender` as well. Thus, disabling receive diversity in the uplink direction will require an additional mechanism to filter all frames received by the secondary AP.

Because the client needs to support receive diversity for MRD-RD and MRD-TRD experiments, we had to install on the laptop client an additional wireless interface as a passive radio. Although it is possible to stack two interfaces next to each other in the laptop's PC-Card slots, such a configuration could potentially generate noise and crosstalk between the interfaces' radio chain. Instead, we installed the second wireless interface in the laptop's internal miniPCI slot (Figure 5.3.1). We then encountered a problem with the passive interface receiving substantially fewer total frames than the primary interface. There were two causes that contributed to the problem. First, the IBM T30 laptop's internal antenna with an impedance that is specifically designed for 802.11b/g interfaces that operated in the 2.4GHz band. Because of mismatched impedance, this type of antenna would attenuate the signals transmitted in 802.11a's 5GHz band. Second, our tests show that MADWiFi's hardware abstraction layer increases the minimum received signal strength threshold (also known as the receive sensitivity threshold) when the interface operates in the 802.11 Monitor mode. The interface filters frames received with a signal strength below the sensitivity threshold. Thus, the increased sensitivity threshold caused the interface to reject more frames.

We used two measures to solve the two problems above. First, we attached a 5dBi gain 2.4GHz external antenna to the miniPCI interface as shown in Figure5.3.1. Although we still used a 2.4GHz antenna (because that was what we had available at the time of the experiments), the external antenna performed substantially better than the internal one. Second, we patched the driver to override the default threshold in Monitor mode. The combination of these fixes helped the passive radio to receive a comparable number of frames to the primary radio. However, we still noticed that the fraction of those frames that are received erroneously is still consistently ($\approx 10\% - 20\%$) higher than the primary radio across all of our MRD-RD and MRD-TRD experiments.

The RDSs in both the RD and TRD experiments use an identical set of MRD-RD parameter values listed in Section 3.6.1 for delivering downlink UDP packets, except we use a MRD-ACK delay of $D = 4$ ms in the new implementation. The only type of frames delivered in the uplink direction are MRD-ACKs, which are not retransmitted if not received by the RDS.

The TDSs in both the TD and TRD experiments use $H = 16$ and $T = 12$ for path selection. We tested several H and T values and found that the increased threshold values (from $H = 4$ and $T = 2$ in the MRD-TD experiments in Chapter 4) tend to perform better on our new testbed. We have not found any compelling reasons that explain why the new values work better. Determining how to tune the H and T parameters that give the best results for MRD-TRD is an open topic for future investigation.

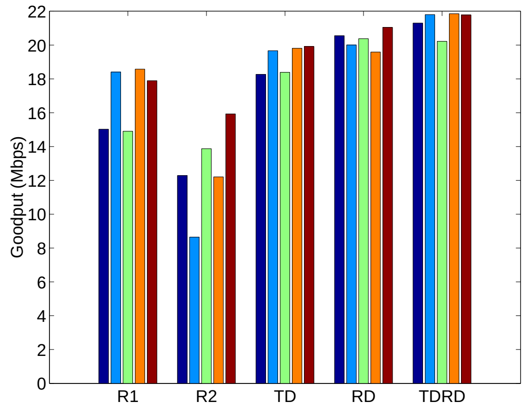
5.3.2 Throughput

Figure 5-6(a) shows the throughput of each trial of our experiment. The average throughput over five trials for the single-radio experiments R1 and R2 were 16.8 Mbps and 12.1 Mbps. Although the node locations are nearly identical[8] to the ones used in the uplink experiments in Section 3.6.1, the throughput of the single-radio downlink experiments is about $2\times$ greater than the corresponding experiments in the uplink direction. Thus, the wireless channel appears to be fairly asymmetric in our experiments.

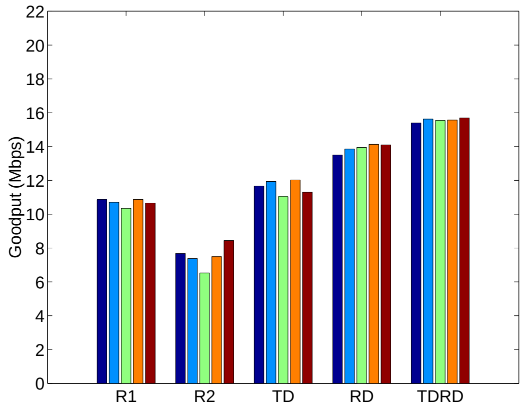
The throughput averages for the TD, RD, TRD experiments are 19.2 Mbps, 20.3 Mbps, and 21.4 Mbps respectively, representing improvement factors of $1.14\times$, $1.21\times$, and $1.27\times$. The gains are substantial but not as impressive as the 1.9-3.0$\times$ improvements that we observed in the uplink experiments. We attribute some of the reduced gains to the signal attenuation caused by the mismatched external antenna on the client's passive radio. Another reason for reduced relative gain might be the increased base-line reference throughput from the single-radio experiments, which decreases the maximum potential gain for any MRD scheme. To test the latter hypothesis, we reduced the transmit power[9] of the radios at R1 and R2 to reduce their base-line throughput and repeated the experiments.

[8]The height of the R2 antenna in the downlink experiments in this chapter is approximately 30 cm lower than the uplink experiments in Chapter 3. Otherwise, the physical locations of the nodes are identical across the downlink and uplink experiments.

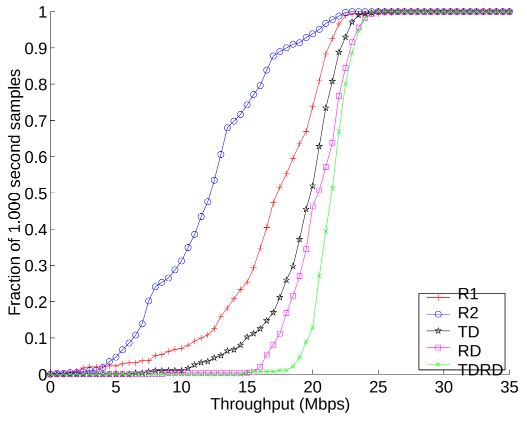
[9]The transmit power reduced from a maximum value of 60 to 6 in the interface's register but we have no documentation about how these values relates to the actual transmit power.



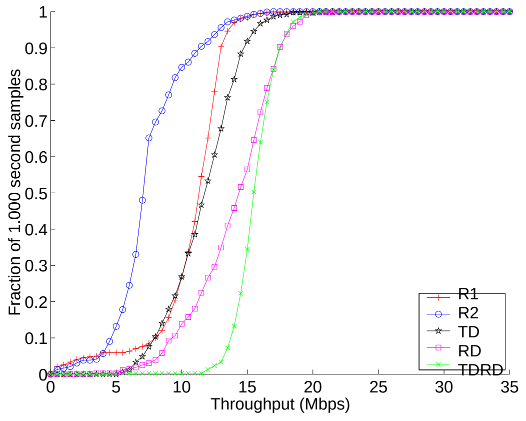
(a) Throughput (Full Tx Power)



(b) Throughput (Low Tx Power)



(c) One-second throughput distribution (Full Tx Power)



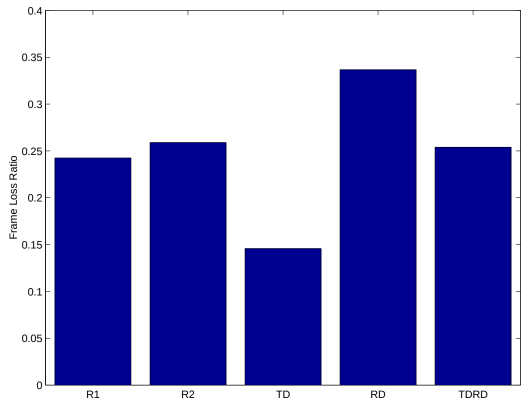
(d) One-second throughput distribution (Low Tx Power)

Figure 5-6: Throughput Analysis (downlink). Top: Throughput of single-path, MRD-TD, MRD-RD, and MRD-TRD experiments; throughput of each trial is represented by a bar within an experiment set. Bottom: Distribution of throughput averaged over non-overlapping one-second window samples. The bottom set of two graphs came from experiments that used a low transmit power setting.

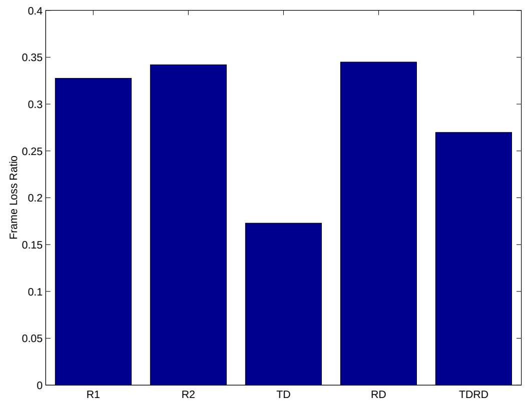
As expected, the reduced transmit power reduced the throughput of all experiments by 27% to 40%, as shown in Figure 5-6(b). The improvement factors of TD, RD, and TRD over R1 in the new set of experiments changed to 1.08×, 1.30×, 1.46×. The next section reveals an implementation bug that could have contributed to the reduced throughput improvements for the downlink experiments. Interestingly, the relative gains over R1 decreased for TD but increased with RD and TRD.

Overall, the relative gains of TRD over TD and over RD are 1.11× and 1.05× in the full transmit power experiments; 1.34× and 1.12× in the low transmit power experiments. Hence, *the integrated MRD-TRD system does provide gains over the individual MRD subsystems*.

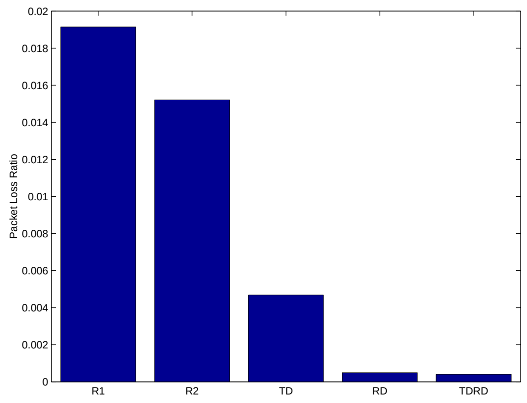
We plot the throughput distribution of the one-second non-overlapping window samples for both full and low transmit power experiments in Figures 5-6(c) and 5-6(d). Not only does the plot shows TRD has higher throughput samples than all other communication



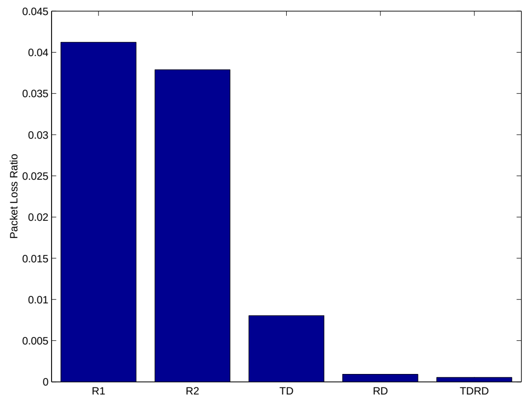
(a) Frame loss rates (Full Tx Pow.)



(b) Frame loss rates (Low Tx Pow.)



(c) Packet loss rates (Full Tx Pow.)



(d) Packet loss rates (Low Tx Pow.)

Figure 5-7: Frame loss rates FLR (Top) and packet loss rates PLR (Bottom) averaged over all five trials of all downlink experiments using low transmit power.

schemes, it also shows substantially lower throughput variation than the rest of the schemes.

5.3.3 Loss Analysis

We plot the average frame loss rates (FLR) and packet loss rates (PLR) for all of the downlink experiment in Figure 5.3.3. For the RD and TRD experiments, FLR is defined to be the number of transmitted frames for which the RDS did not receive a link layer ACK from the receiver's active radio. PLR is defined to be the number of packets that exhausted their retransmission limit and were not delivered successfully to the receiver.

All FLRs are above 14% because our rate adaptation algorithm uses a set of aggressive minimum delivery thresholds to improve throughput (Section 3.4). On the other hand, the TD experiment produced the lowest FLRs among all experiments, because the fine-grained path selection algorithm aims to reduce frame losses. RD had the highest FLRs because the RDS maintains a high bit-rate despite high levels of frame losses in the link between the transmitter and the primary receive radio, as long as the RDC can recover enough frames from soft site selection or frame combining. Not surprisingly, the FLRs of the TRD experiments lie between the FLRs of TD and RD. The results show that the TDS can work

(a) Frame loss/recovery rates (Full Tx Pow.)

Exp.	FLR	FRR	FRR_{SS}	FRR_{FC}
RD	0.283	0.492	0.400	0.092
TRD	0.229	0.424	0.326	0.097

(b) Frame loss/recovery rates (Low Tx Pow.)

Exp.	FLR	FRR	FRR_{SS}	FRR_{FC}
RD	0.331	0.500	0.463	0.037
TRD	0.260	0.382	0.340	0.043

Table 5.1: Frame loss (FLR) and frame recovery rates (FRR) averaged over five trials of low transmission power and high channel variability downlink experiments. FRR is decomposed into two sources of recovery: soft selection (FRR_{SS}) and frame combining (FRR_{FC}).

in conjunction with the RDS to help reduce frame loss rates and as a result, increase the overall throughput.

The PLR of the R1 and R2 experiments are about four folds higher than TD and over an order of magnitude higher than the RD and TRD experiments. One reason that explains is that losses occur in bursts more often in the single-path schemes than the schemes that use path diversity. Thus, R1 and R2 have proportionately more packets that exhaust their retransmission limit than the MRD schemes. The absolute PLRs (1.5 - 4.1%) of R1 and R2 are "small" but these packet loss levels can cause serious throughput degradations in TCP **??**.

However, we later uncovered a timeout bug in the implementation that prevented the transmitter to retransmit a frame up to the set limit of 7 (excluding the first transmission of a frame) in the single-radio experiments. The timeout bug was clearing frames from the sender's buffer prematurely and reduced the effective retransmission limit to about 2 or 3 for each frame. As a result, the bug could have increased the packet loss rate. At the same time, the bug could also have increased the measured throughput for the single-radio schemes in the previous section because the bug essentially reduces the upper limit of the total number of retransmission failures for each lost packet. Because the timeout bug affected only the single-radio receiver schemes, the bug might have also impacted the observed improvement of MRD-TRD over these schemes.

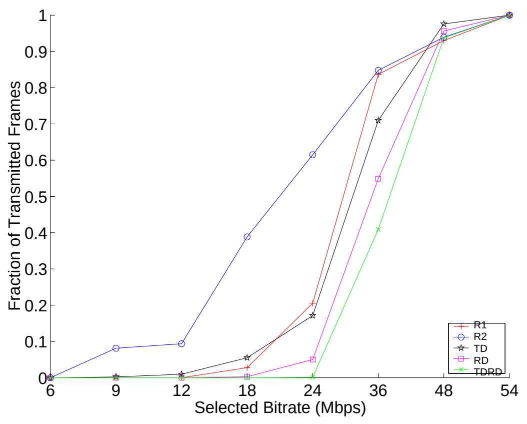
5.3.4 Error Recovery Rates

As we learned from the previous uplink experiments in Section 3.6.2, the performance of the receive diversity sub-system depends on its ability to recover frame losses from site selection (*i.e.,* those frames correctly received by the passive radio at the receiver) and from frame combining.

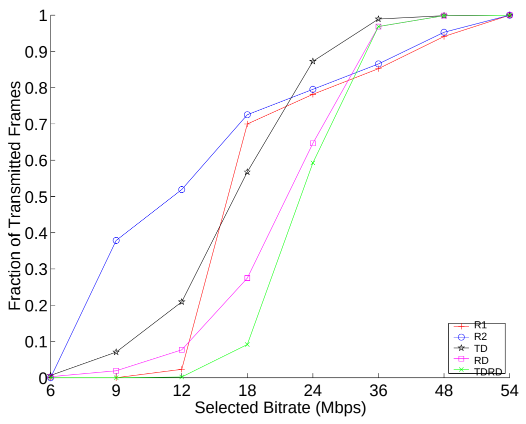
Table 5.3.4 shows the frame loss and the decomposed recovery rates for the downlink RD and TRD experiments. In the experiments that used full transmit power, the FRR values are comparable to the values measured for the uplink experiments.

5.3.5 Transmission bit-rates

The performance of every communication depends on the bit-rates that were selected for frame transmissions as well. Figures 5-8(a) and 5-8(b) show that the TDR experiments had a higher and narrower transmission bit-rate distribution than all of the other experiments, which help produce high and consistent measured throughput in our experiments.

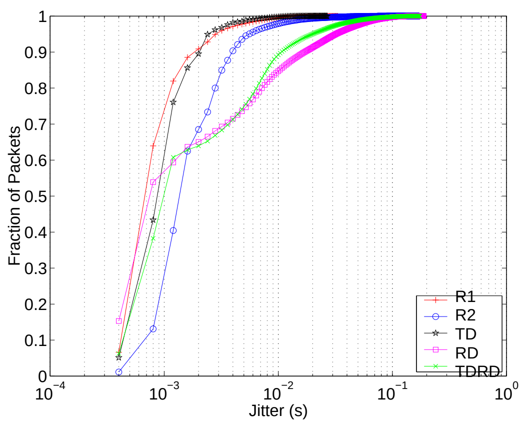


(a) CDF of selected bit-rate (Full Tx Pow.)

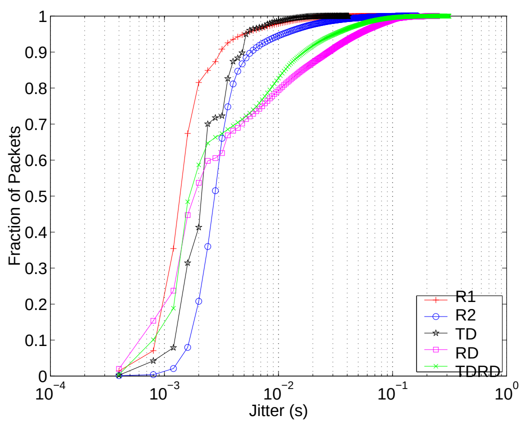


(b) CDF of selected bit-rate (Low Tx Pow.)

Figure 5-8: Distribution of selected bit-rate for each downlink frame transmission.



(a) Jitter CDF (Full Tx Pow.)



(b) Jitter CDF (Low Tx Pow.)

Figure 5-9: One way delay jitter for downlink experiments using low transmit power.

5.3.6 Delay Analysis

We use the method described in Section 3.6.2 to compute the one-way delays for each delivered packet and plot the resulting distribution in Figure 3-15. The shape of the delay distributions of the R1, R2, RD experiments are similar to the ones for the uplink experiments presented in Section 3.6.2. The TRD experiments had slightly lower delays than RD because TRD had lower frame loss rates and therefore, required fewer retransmissions than RD.

In contrast to the conclusion drawn in Section 4.4.3, the TD experiments did not reduce delays over the single-radio experiments of R1, even though TD was able to reduce a great number of frame losses over R1 as shown in the previous section. We attribute the delay to the interaction between retransmissions in MRD-TD and the frame transmission pipeline. Unlike in R1 and R2, where retransmissions are performed locally at the transmitter, retransmissions in the MRD-TD scheme are performed by the TDS running over the backbone infrastructure. Thus, the actual retransmission might not occur until after a round-trip delay plus some number of transmission delays of frames that are queued in the transmission pipeline. The process of filling an empty transmission pipeline, which happens whenever the TDS switches transmission paths, also adds a small amount of delay in delivering a packet.

5.4 Summary

This chapter covers the design of MRD-TRD, a system that integrates the MRD-RD and MRD-TD sub-systems described in the previous two chapters to provide gains of both receive and transmit diversity in both uplink and downlink directions of data delivery. We provide a clean implementation of the system, which integrates the sender-side components (the RDS and TDS) into a single user-level process that can run on a central router and forward frames over the network to/from a set of diversity radios.

To enhance the efficiency of the system, we introduced four optimizations in the MRD-TRD system. These include:

- Pipelined transmissions, which helps improve throughput in presence of feedback delays in the backbone network;
- Transmit sequence numbering, which helps the RDS to identify stale feedback from the RDC
- Early retransmits, which helps keep the channel utilized when not busy and helps prevent traffic (especially TCP) from stalling;
- Out-of-order packet delivery, which reduces latency for applications that do not require ordered packet delivery

We measured the downlink UDP transmission performance of MRD-TRD and compared it against MRD-RD and MRD-TD. Our results show that MRD-TRD provides up to 34% and 12% throughput gains over MRD-TD and MRD-RD respectively. Furthermore, MRD-TRD exhibits much less throughput variation than the MRD-RD and MRD-TD schemes.

For future work, it would be interesting to examine TCP performance, how the performance change as we add more radios at the transmitter and receiver, and evaluate how MRD-TRD performs in a scenario that consists of many contenders.

Chapter 6

Conclusion and Future Work

This chapter provides a summary of this dissertation and discusses directions for future work.

6.1 Summary

In this dissertation, we addressed the challenge of improving performance and packet delivery efficiency in wireless LANs. We showed how channel variabilities in the wireless medium can lead to high frame loss rates and clustered loss patterns. They cause inefficient packet delivery in current wireless WANs because the transmitter can retransmit wastefully or select a conservatively low bit-rate to compensate for losses. As a result, connections often suffer from low throughput, increased packet loss rates, and inconsistent performance.

Although diversity is a commonly known technique used to mitigate channel variations in communications theory, we observe that existing wireless LANs adopts a model that treats each terminal as an independent entity and do not take advantage of the *path diversity* that inherently exists among nearby APs or radios in the system's infrastructure. To this end, we developed the Multi-Radio Diversity system, which includes the following key components to help improve performance in a wireless LAN:

- **MRD-Receive Diversity (MRD-RD).** This sub-system allows a wireless LAN to reduce losses by using multiple radio receivers. It funnels all received versions of the same transmission from different radios or APs to a central controller, so that it can forward onto the rest of the network an error-free version of the frame while filtering the redundant ones. We introduced a practical block-based *frame combining* algorithm, a technique that can produce an error-free version of a frame even when none of the receivers manage to receive an error-free version of the original transmission. For efficient retransmissions, MRD-RD incorporates the *request-for-acknowledgment* (RFA) protocol. MRD-RD uses RFA to retrieve from the central controller the final reception status of a given transmission but maintains high throughput by not blocking subsequent transmissions immediately after a transmission error. In addition, MRD-RD uses an enhanced rate adaptation algorithm that adjusts transmission bit-rates using the losses observed at the central controller.

- **MRD-Transmit Diversity (MRD-TD).** This sub-system allows a wireless LAN to reduce losses by using multiple transmitters. It uses spoofing to decouple the process of associating a client with an AP from the process of delivering data frames to the client to facilitate low-overhead path switching, and incorporates a practical fine-grained path selection heuristic that effectively reduces burst losses in environments with high channel variations.

- **MRD-Transmit and Receive Diversity (MRD-TRD).** This system integrates both MRD-RD and MRD-TD to combine the gains of receive and transmit diversity and further increase the efficiency of packet delivery in wireless LANs.

In addition, this dissertation demonstrates two key results:

- **MRD helps reduce losses and improves efficiency and throughput without consuming much wireless bandwidth.** We have conducted a wide variety indoor measurements that show packet loss rate reductions of up to 75% and throughput improvements of up to 3×. Even though MRD does not aggregate bandwidth over multiple radio frequencies, our measurements show that MRD can sometimes even achieve higher throughput than bandwidth aggregation.

- **Reducing loss variance improves the efficiency of a simple rate adaptation algorithm.** While the conventional wisdom in managing link quality is to have the sender adapt the bit-rates first *before* attempting a handoff (using another path), MRD takes an opposite approach that allows a sender to communicate over multiple paths first, *then* adapt the bit-rate. MRD exploits path diversity to reduce both frame losses and loss variations. Thus, even a simple rate adaptation algorithm, such as the one we used, can maintain high throughput, despite high losses and high variations in the individual paths.

6.2 Future Work

The MRD wireless architecture presents many interesting areas that can be further developed and greatly improved. We describe a few of them below.

Scalability. One of the issues we have not addressed in this dissertation is scaling MRD-enabled wireless LANs to tens and hundreds of APs, while coping with mobility. The system needs a way to manage load on MRD's central controller and traffic on the backbone network as the number of APs increase in the network. One possibility is to scale the central controller using a cluster of servers and implement a classifier that balances load by distributing traffic from different clients to a different server. Another possibility is to partition the network into smaller network segments and deploy a central controller to manage the traffic for each segment. In addition to traditional mobility problems such as authenticating and managing IP address assignments for a mobile client moving across segment boundaries, the problem of migrating MRD state between the central controllers without disrupting the flow of the clients' traffic must be solved.

Another research direction for improving scalability is to develop mechanisms for reducing traffic load on the backbone network. Currently, the receive diversity sub-system forwards *all* received version of the same transmission to the central controller. Thus, if N APs are within reception range of a client, the traffic load on the backbone increases by up to a factor of N. One key observation is that the bulk of the traffic is redundant information that can be filtered out by either snooping on the wireless medium (*e.g.,* do not forward a client's transmission to the central controller if the passive AP detects a link layer ACK from the active AP) or snooping on the backbone (*e.g.,* do not forward a client's transmission if another AP has already forwarded an error-free reception of the frame). To support an infrastructure that is interconnected over a wireless backbone, we might even adapt a more sophisticated reception reporting scheme that is used in the ExOR protocol [25] to help reduce load on the backbone.

Deprecating the Handoff Process. One of the contributions of MRD's system design is removing the constraint that a sender can only communicate with one AP (or radio) at a time. However, the current MRD design still requires a client to associate with a primary (active) AP and to perform handoffs between primary APs, which could disrupt the flow of the clients' traffic. An open question is whether we can replace the handoff process completely by extending the function of the RDC to manage client mobility. We can configure the client to periodically send a broadcast message and use the RDC to determine the set of APs that could be used to communicate with the client. The RDC can then choose the most appropriate AP to be the primary AP (to be responsible for sending link layer ACKs for uplink transmissions) for a given client at a given location. In such a system, the client should not need to disrupt traffic flow for scanning and discovering unknown APs as it moves within the network.

Dynamic Role and Channel Assignment. An extension of MRD is the notion of dynamic channel assignment and using the technique to help balance load and improve capacity in the network. The current MRD design assumes a cellular frequency reuse model where the operating radio frequency is statically assigned to each primary (active) AP. As a result, the roles of primary and secondary APs (used to provide alternative transmission and reception paths) are also statically assigned for each AP (or for each co-located radio in a single AP) within a cell.

Static radio frequency and role assignments can be sub-optimal in many scenarios. For example, many wireless clients can be clustered within a conference room covered by a single cell. Instead of assigning a single radio frequency for that cell, the system ought to be able to dynamically assign multiple operating frequencies to that cell and balance load by re-distributing the clients to operate at the different frequencies. Dynamic frequency assignment is a classical problem for cellular phone networks [].XXX Similarly, the system should be able to assign different (primary and secondary) roles to each AP so that the system can strike a balance between the number of operating frequencies that can be supported for a given area (*i.e.,* the number of APs designated as primary) and coverage (*i.e.,* the number of APs designated as secondary). Solving this problem will require a better understanding about how capacity and achieved throughput are related, and developing a model that predicts how throughput gains from path diversity trades off with gains from aggregating multiple channels. We alluded to some of these issues in Section 3.7.2.

Exploiting Capture Effect. When two or more senders transmit simultaneously, one of the transmissions could capture the receiver to produce a successful reception despite interference from other ongoing transmissions. The success of capture depends on the modulation scheme and the relative signal strengths of the arriving signals that the receiver observes. A recent measurement study shows that capture effects can be significant in networks based on 802.11g [47].

Because MRD uses multiple receive radios, there could be multiple opportunities for receiving captured transmissions. It would be interesting to quantify and measure the gains of capture effects in the MRD system. Moreover, the capture properties are likely to be different at different receivers because of path diversity. While one transmitter might capture the reception at one receiver, another transmitter might capture a receiver at a different location. Thus, with multiple receive radios, the receiver-side of the system could potentially be used to 1) identify capture events when they occur and 2) determine which (subset of) transmitters were involved in the capture event. Determining how to exploit this information to improve performance is an interesting open topic.

Other Improvements. We have also outlined several other kinds of improvements throughout Chapters 3-5. These include:

- Modifying the Medium Access Control protocol to use an idle-sense mechanism [43, 86] or become MRD-aware to avoid excessive congestion window back-offs (Section 3.7.3).
- Enhancing the fine-grained selection algorithm by adapting the H and T thresholds based on relative long-term channel quality in each transmission path (Section 4.2.1).
- Refining the frame combining algorithm by incorporating an incremental CRC computation scheme [27, 81] to reduce the running time and hence, permitting the MRD to run a stronger frame combining process (Section 3.6.2).
- Exploring and evaluating various rate adaptation schemes and other optimizations described in Sections 3.6.2, 3.7.1 and 5.1.1 to improve MRD's throughput and delay performance.

Despite the laundry list of potential improvements, we have shown that the current MRD system can improve throughput by up to a factor of 3× over conventional wireless LANs.

Bibliography

[1] CRAWDAD: A community resource for archiving wireless data at dartmouth. http://crawdad.cs.dartmouth.edu/.

[2] Feasibility study for enhanced uplink for UTRA FDD. 3GPP Technical Report TR 25.896 v6.0.0.

[3] Madwifi: Multiband Atheros Driver for WiFi. http://madwifi.sourceforge.net/.

[4] tcpdump/libpcap. http://www.tcpdump.org.

[5] TinyOS: An open-source OS for the networked sensor regime. http://www.tinyos.net.

[6] UTRA high speed downlink packet access (HSUPA). 3GPP Technical Report TR 25.950 v4.0.0.

[7] Engim product overview. http://www.engim.com/products.html, 2003.

[8] IEEE P802.11i/D10.0. Medium Access Control (MAC) Security Enhancements, April 2004.

[9] Madwifi mailing list archive. http://news.gmane.org/gmane.linux.drivers.madwifi.devel, February 2005.

[10] Wi-Fi chipset fever: 140 million and growing. In-Stat Research (IN0501813NT), December 2005.

[11] A2475 embeded 802.11b/g smart antenna. http://www.airgain.com/A2475.html, 2006.

[12] Beamflex. http://www.ruckuswireless.com/technology/beamflex.php, 2006.

[13] Meru networks - radio switch. http://www.merunetworks.com/products/radio_switch.shtml, 2006.

[14] IEEE 802.11a Medium Access Control (MAC) and Physical Layer (PHY) Specification: High-speed Physical Layer in the 5 GHz Band , December 1999.

[15] IEEE 802.11b/d3.0 Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, August 1999.

[16] IEEE 802.11g Medium Access Control (MAC) and Physical Layer (PHY) Specification Amendment 4: Further High Data Rate Extension in the 2.4 GHz Band , June 2003.

[17] IEEE Standards for Local and Metropolitan Area Networks: Standard for Port Based Network Access Control, October 2001.

[18] M. G. Arranz, R. Aguero, L. Munoz, and P. Mahonen. Behavior of UDP-based applications over IEEE 802.11 wireless networks. In *Proc. of IEEE PIMRC*, volume 2, pages 72–77, San Diego, CA, September 2001.

[19] Arun Avudainayagam, John Shea, Tan Wong, and Xin Li. Reliability exchange schemes for iterative packet combining in distributed arrays. In *Proc. of IEEE WCNC*, pages 832–837, New Orleans, LA, March 2003.

[20] Paramvir Bahl, Atul Adya, Jitendra Padhye, and Alec Wolman. Reconsidering wireless systems with multiple radios. *ACM CCR*, pages 39–46, October 2004.

[21] Paramvir Bahl, Ranveer Chandra, Jitendra Padhye, Lenin Ravindranath, Manpreet Singh, Alec Wolman, and Brian Zill. Enhancing the security of corporate Wi-Fi networks using DAIR. In *In Proc. of ACM/USENIX MobiSys*, Upsalla, Sweden, June 2006.

[22] Dimitri Bertsekas and Robert Gallager. *Data Networks.* Prentice Hall, second edition, 1992.

[23] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. Macaw: a media access protocol for wireless lan's. pages 212–225, London, United Kingdom, August 1994.

[24] John C. Bicket. Bit-rate selection in wireless networks. Master's thesis, Massachusetts Intitute of Technology, Cambridge, MA, February 2005.

[25] Sanjit Biswas and Robert Morris. Exor: opportunistic multi-hop routing for wireless networks. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 133–144, New York, NY, USA, 2005. ACM Press.

[26] Specification of the bluetooth system. `http://www.bluetooth.com/`, December 1999. Bluetooth Special Interest Group document.

[27] Florian Braun and Marcel Waldvogel. Fast incremental CRC updates for IP over ATM networks. In *Proc. of IEEE HPSR*, Dallas, TX, May 2001.

[28] T. Camp, J. Lusth, J. Matocha, and C. Perkins. Reduced cell switching in a mobile computing environment. In *Proc. of ACM MobiCom*, Boston, MA, August 2000.

[29] S. S. Chakraborty, M. Lilnaharja, and K. Ruttik. Diversity and packet combining in rayleigh fading channels. *IEE Proceedings - Communications*, 152:353–356, June 2005.

[30] Shyam Chakraborty, Erkki Yli-Juuti, and Markku Liinaharja. An ARQ scheme with packet combining. *IEEE Communications Letters*, 2(7):200–202, July 1998.

[31] Ranveer Chandra, Victor Bahl, and Pradeep Bahl. MultiNet: Connecting to Multiple IEEE 802.11 Networks Using a Single Wireless Card. In *infocom2004*, 2004.

[32] David Chase. Code combining—a maximum-likelihood decoding approach for combining an arbitrary number of noisy packets. *IEEE Transactions on Communications*, 33(5), May 1985.

[33] Sunghyun Choi, Youngkyu Choi, and Inkyu Lee. IEEE 802.11 MAC-level FEC with retransmission combining. *IEEE Transactions on Wireless Communication*, January 2005.

[34] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. In *Proc. of ACM MobiCom*, San Diego, CA, September 2003.

[35] Abdel-Ghani A. Daraiseh and Carl W. Baum. Methods for packet combining in HARQ systems over bursty channels. *Mobile Networks and Applications*, 2:213–224, October 1997.

[36] R. T. Derryberry, D. M. Ionescu S. Gray, G. Mandyam, and B. Raghothaman. Transmit diversity in 3G CDMA systems. *IEEE Communication Magazine*, pages 68–74, 2002.

[37] Suhas Diggavi, Naofal Al-Dhahir, A. Stamoulis, and A. R. Calderbank. Great expectations: The value of spatial diversity in wireless networks. (2):217–270, February 2004.

[38] Henri Dubois-Ferrière, Deborah Estrin, and Martin Vetterli. Packet Combining in Sensor Networks. In *3rd ACM Conference on Embedded Networked Sensor Systems (SenSys 2005)*, 2005.

[39] David Eckhardt and Peter Steenkiste. Measurement and analysis of the error characteristics of an in-building wireless network. In *Proc. of ACM SIGCOMM*, pages 243–254, Palo Alto, CA, August 1996.

[40] H. Furukawa, K. Hamabe, and A. Ushirokawa. SSDT–site selection diversity transmission power control for cdma forward link. *IEEE JSAC*, 18:1546–1554, 2000.

[41] James Gross and Andreas Willig. Measurements of a wireless link in different RF-isolated environments. In *Proc. of European Wireless 2002*, Florence, Italy, February 2002.

[42] Juha Heiskala and John Terry. *OFDM Wireless LANs: A Theoretical and Practical Guide.* Sams Publishing, Indianapolis, IN, 2002.

[43] Martin Heusse, Franck Rousseau, Romaric Guillier, and Andrzej Duda. Idle sense: An optimal access method for high throughput and fairness in rate diverse wireless LANs. In *Proc. of SIGCOMM'05*, pages 121–132, Philadelphia, PA, August 2005.

[44] Gavin Holland, Nitin H. Vaidya, and Paramvir Bahl. A rate-adaptive MAC protocol for multi-hop wireless networks. In *Proc. of ACM MobiCom*, pages 236–251, Rome, Italy, July 2001.

[45] Harri Holma and Antti Toskala. *WCDMA for UMTS: Radio Access for Third Generation Mobile Communications (3rd Edition).* John Wiley & Sons, Ltd., West Sussex, England, 2004.

[46] IEEE 802.11 Working Group. Draft Supplement to International Standard for Information Exchange between systems - LAN/MAN Specific Requirements, November 2001.

[47] Kyle Jamieson, Bret Hull, Allen Miu, and Hari Balakrishnan. Understanding the real-world performance of carrier sense. In *E-WIND '05: Proceeding of the 2005 ACM SIGCOMM workshop on Experimental approaches to wireless network design and analysis*, pages 52–57, New York, NY, USA, 2005. ACM Press.

[48] Zhengrong Ji, Yi Yang, Junlan Zhou, Mineo Takai, and Rajive Bagrodia. Exploiting medium access diversity in rate adaptive wireless LANs. In *Proc. of ACM MobiCom*, pages 345–359, Philadelphia, PA, September 2004.

[49] Wenyu Jiang. Bit error correction without redundant data: A MAC layer technique for 802.11 networks. In *Proceedings of the Second Workshop on Wireless Network Measurements (WiNMee 2006)*, Boston, MA, USA, April 2006.

[50] Jouni Malinen. Host AP driver for Intersil Prism2/2.5/3. `http://hostap.epitest.fi/`, 2003. Version 0.0.1.

[51] A. Kamerman and L. Monteban. Wavelan ii: A high-performance wireless lan for the unlicensed band. *Bell Labs Technical Journal*, pages 118–133, Summer 1997.

[52] Sachin Katti, Dina Katabi, Wenjun Hu, Hariharan Rahul, and Muriel Medard. The importance of being opportunistic: Practical network coding for wireless environments. In *In Proc. of 43rd Allerton Conference on Communication, Control, and Computing*, Allerton, IL, September 2005.

[53] R. Knopp and P. A. Humblet. Information capacity and power control in single-cell multiuser communications. In *Proc. of IEEE ICC*, pages 331–335, Seattle, WA, June 1995.

[54] Andreas Kopsel and Adam Wolisz. Voice transmission in an IEEE 802.11 WLAN based access network. In *Proc. of ACM WoWMoM*, pages 23–32, Rome, Italy, July 2001.

[55] Mathieu Lacage, Mohammad H. Manshaei, and Thierry Turietti. IEEE 802.11 rate adaptation: A practical approach. In *Proc. of ACM MSWiM*, pages 126–134, Venezia, Italy, October 2004.

[56] Nicholas Laneman. *Cooperative Diversity in Wireless Networks: Algorithms and Architectures.* PhD thesis, Massachusetts Institute of Technology, August 2006.

[57] V. C. M. Leung and A. W. Y. Au. A wireless local area network employing distributed radio bridges. *Wireless Networks*, 2:97–107, 1996.

[58] Yong Liang and Shyam S. Chakraborty. ARQ and packet combining with post-reception selection diversity. In *Proc. of IEEE VTC'04*, Los Angeles, CA, September 2004.

[59] Shu Lin, Daniel Costello, and Michael Miller. Automatic-repeat-request error-control schemes. *IEEE Communications Magazine*, 22(12):5–17, December 1984.

[60] Shu Lin and Daniel J. Costello. *Error Control Coding: Fundamentals and Applications.* Prentice Hall, second edition, 2004.

[61] Shu Lin and Daniel J. Costello Jr. *Error Control Coding : Fundamentals and Applications.* Prentice Hall, 1983.

[62] Shu Lin, Daniel J. Costello Jr., and Michael J. Miller. Automatic-repeat-request error control schemes. *IEEE Communications Magazine*, (12):5–17, December 1984.

[63] Arunesh Mishra, Minho Shin, and William Arbaugh. An empirical analysis of the IEEE 802.11 MAC layer handoff process. Technical Report 75, University of Maryland, 2002.

[64] Vlad Mitlin. Optimal MAC packet size in networks without cut-through routing. *IEEE Transactions on Wireless Communications*, 2(5):901–910, September 2003.

[65] Vlad Mitlin. Optimal selection of ARQ parameters in QAM channels: Research articles. *Wireless Communicaitons and Mobile Computing*, 5(2):165–174, 2005.

[66] Allen Miu, John Apostolopoulos, Wai T. Tan, and Mitch Trott. Low-latency wireless video over 802.11 networks using path diversity. In *Proc. of IEEE ICME*, volume 2, pages 441–444, Baltimore, MD, July 2003.

[67] Allen Miu, Hari Balakrishnan, and Can Emre Koksal. Multi-radio diversity in wireless networks. In *Proc. of ACM MobiCom*, Cologne, Germany, September 2005.

[68] Allen Miu, Godfrey Tan, Hari Balakrishnan, and John Apostolopoulos. Divert: Fine-grained path selection for wireless LANs. In *Proc. of ACM MobiSys*, pages 203–216, Boston, MA, June 2004.

[69] E. Modiano. An adaptive algorithm for optimizing the packet size used in wireless ARQ protocols. *Wireless Networks*, pages 279–286, 1999.

[70] Andreas F. Molisch and Moe Z. Win. MIMO systems with antenna selection. *IEEE Microwave Magazine*, pages 46–56, March 2004.

[71] C. Monteleoni and T. Jaakkola. Online learning of non-stationary sequences. In *Proc. of Advances in Neural Information Processing Systems*, 2003.

[72] Sue B. Moon, Paul Skelly, and Don Towsley. Estimation and removal of clock skew from network delay measurements. In *Proc. of IEEE INFOCOM*, volume 1, pages 227–234, New York, NY, March 1999.

[73] Konstantina Papagiannaki, Mark Yarvis, and W. Steven Conner. Experimental characterization of home wireless networks and design implications. In *Proc. of IEEE INFOCOM*, Barcelona, Spain, April 2006.

[74] Larry Peterson and Bruce Davie. *Computer Networks: A Systems Approach.* Morgan Kaufmann Publishers, San Francisco, CA, third edition, 2003.

[75] D. Phatak and T. Goff. A novel mechanism for data streaming across mutiple ip lnks for improving throughput and reliability in mobile environments. New York, NY, June 2002.

[76] D. Qiao and S. Choi. Goodput enhancement of IEEE 802.11a wireless LAN via link adaptation. In *Proc. of IEEE ICC*, pages 161–175, Helsinki, Finland, June 2001.

[77] Asfandyar Qureshi and John Guttag. Horde: separating network striping policy from mechanism. In *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 121–134, New York, NY, USA, 2005. ACM Press.

[78] T.S. Rappaport. *Wireless Communications.* Prentice Hall, Upper Saddle River, N.J., 1996.

[79] Pablo Rodriguez, Rajiv Chakravorty, Ian Pratt, and Suman Banerjee. MARS: A commuter router infrastructure for the mobile internet. In *Proc. of ACM MobiSys*, Boston, MA, June 2004.

[80] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly. Opportunistic media access for multirate ad hoc networks. In *Proc. of ACM MobiCom*, pages 24–35, Atlanta, GA, September 2002.

[81] Julian Satran, Dafna Sheinwald, and Ilan Shimony. Out of order incremental CRC computation. *IEEE Transactions on Computers*, pages 1178–1181, September 2005.

[82] Minho Shin, Arunesh Mishra, and William A. Arbaugh. Improving the latency of 802.11 hand-offs using neighbor graphs. In *Proc. of ACM MobiSys*, Boston, MA, June 2004. ACM Press.

[83] Pradeep Sindhu. Retransmission error control with memory. *IEEE Trans. on Communications*, 25:473–479, May 1977.

[84] A. Snoeren. Adaptive inverse multiplexing for wide-area wireless networks. pages 1665–1672, December 1999.

[85] Richard W. Stevens. *TCP/IP Illustrated.* Addison-Wesley, Reading, MA, 1994.

[86] Godfrey Tan. *Improving aggregate user utilities and providing fairness in multi-rate wireless LANs.* PhD thesis, Massachusetts Institute of Technology, February 2006.

[87] Godfrey Tan and John Guttag. Time-based fairness improves performance in multi-rate WLANs. Boston, MA, June 2004.

[88] A. Tanenbaum. *Computer Networks.* Prentice-Hall, Englewood Cliffs, N.J., 1996.

[89] Chiping Tang and Philip K. McKinley. Modeling multicast packet losses in wireless LANs. Technical report, Computer Science and Engineering Department, Michigan State University, May 2003.

[90] Jean Tourrilhes. Fragment adaptive reduction : Coping with various interferers in radio unlicensed bands. In *Proc. of IEEE ICC*, Helsinki, Finland, June 2001.

[91] Matthew C. Valenti. Improving uplink performance by macrodiversity combining packets from adjacent access points. In *Proc. of IEEE WCNC*, pages 636–641, New Orleans, LA, March 2003.

[92] A. Willig, M. Kubisch, C. Hoene, and A. Wolisz. Measurements of a wireless link in an industrial environment using an IEEE 802.11-compliant physical layer. *IEEE Trans. on Industrial Electronics*, 43:1265–1282, December 2002.

[93] James M. Wilson. The next generation of wireless LAN emerges with 802.11n. Device Forge, August 2004. `http://www.deviceforge.com/articles/AT5096801417.html`.

[94] Daniel Wong and Teng Lim. Soft handoffs in CDMA mobile systems. *IEEE Personal Communications Magazine (PCS)*, 4(6):6–17, December 1997.